Nuclear Fuel Management Optimisation Using Estimation of Distribution Algorithms

Submitted to the University of London, UK

for the degree of Doctor of Philosophy

Shan Jiang

Imperial College London

University of London, UK.

August 2009

Declaration of originality

The material in this thesis is entirely the results of my own independent research under the supervision of Dr. Jonathan N. Carter and Prof. Christopher C. Pain, and is not the outcome of any collaborative work. All published or unpublished material used in this thesis has been given full acknowledgement.

_____ _____

Name Date

Publications

Peer Reviewed Publications

S. Jiang, A. K. Ziver, J. N. Carter, C. C. Pain, A. J. H. Goddard, S. J. Franklin and H. Phillips, 2006, Estimation of Distribution Algorithms for Nuclear Reactor Fuel Management Optimisation. Annals of Nuclear Energy, Vol. 33, Issue 11-12, pages 1039-1057.

Conference Proceedings

S. Jiang, A. K. Ziver, J. N. Carter, C. C. Pain, M. D. Eaton, A. J. H. Goddard, S. J. Franklin and H. Phillips 2006, Research Reactor Loading Pattern Optimisation using Estimation of Distribution Algorithms. PHYSOR 2006, September 10-14, Vancouver, BC, Canada.

S. Jiang, C. C. Pain, J. N. Carter, A. K. Ziver, M. D. Eaton, A. J. H. Goddard, S. J. Franklin and H. Phillips, 2008, Nuclear Reactor Reactivity Prediction using Feed Forward Artificial Neural Networks. The Fifth International Symposium on Neural Networks, September 24-28, Beijing, China.

Abstract

In this thesis, Estimation of Distribution Algorithms (EDAs) are used to solve nuclear reactor fuel management optimisation problems. The main goal of nuclear reactor fuel management optimisation is to search for loading patterns (LPs) of fuel assemblies which maximize the performance of the reactor subject to various constraints. It is normally classified as a difficult combinatorial optimisation problem in the field of nuclear engineering.

Similar to typical population based optimisation algorithms, e.g. Genetic Algorithms (GAs), EDAs maintain a population of solutions and evolve them during the optimisation process. Unlike the GAs, new solutions are suggested by sampling the probability distribution model estimated from the selected promising solutions.

Following a background introduction and literature review, we apply the EDAs to a set of classical analytical Travelling Salesman Problems (TSPs) to investigate the EDA's behaviour when solving combinatorial optimisation problems which are similar to the nuclear reactor LP optimisation. Based on comparison study with well established GAs with various crossover operators, a couple of EDAs with performance enhancements are proposed to solve the real-world reactor LP optimisation.

The proposed EDAs are then employed to solve realistic reactor LP optimisation cases derived from the Imperial College CONSORT research reactor. Numerical experiments and comparison studies are carried out with the benchmark GAs. The conclusions are drawn that the proposed EDAs have outperformed the current state-of-the-art GAs on the test cases. The probability model used in the EDAs enables better understanding, analysis and fine-tuning of the algorithm performance. The EDAs are promising candidates to be applied to more complicated reactor fuel management optimisation problems.

Acknowledgments

I would like to thank the following people and organisations:

- My academic supervisors, Dr. J. N. Carter and Prof. C. C. Pain for their consistent guidance, patience and encouragement in last three years. I also want to thank Prof. Pain particularly for partially financing my tuition fee;
- Dr. A. K. Ziver, for his enormous support at the beginning of my study in entering the field of nuclear engineering and his guidance on the CONSORT reactor project and help on the writing of part of the Appendix of this thesis;
- Dr. M. D. Eaton, for his help on the use of the WIMS code and his encouragement on my research that motivates me most in hard times;
- Prof. A. J. H. Goddard, for his partial financial support that has helped me through the most difficult time;
- Dr. J. L. M. A. Gomes, for his help on the use of the in-house code (i.e., FETCH, FLUIDITY and EVENT) and very helpful discussions;
- Mr. S. J. Franklin and Mrs. H. J. Phillips, for agreeing the use and manipulating the data from the CONSORT reactor and their comments on our publications;
- Prof. Cassiano de Oliveira, for make available the use of the EVENT code;
- Everyone in the Applied Modelling and Computation Group (AMCG) at the Earth Science and Engineering Department for their friendship and encouragement;
- The support staff of the Earth Science and Engineering Department at Imperial College London, for their excellent work in supporting of my study;
- Finally, my parents, parents-in-law and my wife, for their support both financially and mentally. I could never have come this far without them.

To my parents, my brother and my wife

Abbreviations

ACO	Ant Colony Optimisation
ANN	Artificial Neural Network
BMDA	Bivariate Marginal Distribution Algorithm
BOA	Bayesian Optimization Algorithm
CANDU	Canada Deuterium Uranium Reactor
CGA	Compact Genetic Algorithm
CX	Cycle Crossover
DE	Differential Evolution
EA	Evolutionary Algorithm
EBNA	Estimation of Bayesian Networks Algorithm
EDA	Estimation of Distribution Algorithm
ES	Evolutionary Strategy
FDA	Factorized Distribution Algorithm
GA	Genetic Algorithms
GP	Genetic Programming
HTBX	Heuristic Tie Breaking Crossover
LP	Loading Pattern
MIMIC	Mutual Information Maximization for Input Clustering
NRFMO	Nuclear Reactor Fuel Management Optimisation
OX	Ordering Crossover
PBIL	Population Based Incremental Learning
PPF	Power Peaking Factor
QAP	Quadratic Assignment Problem
SA	Simulated Annealing
TS	Tabu Search
TSP	Travelling Salesman Problem
UMDA	Univariate Marginal Distribution Algorithms
UX2	Union Crossover # 2

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Main Drive

Nuclear power stations have been an important part of the world's electrical energy generation since the 1950s. According to the World Nuclear Association, WNA (2007), there are now 439 commercial nuclear power reactors operating in 30 countries, with 372,000 MWe of total capacity, providing approximately 16% of the world's electricity. A total of 284 research reactors, in 56 countries, and a further 220 reactors for power ships and submarines are also in use.

Because of the world's increasing energy demand and the critical nature of environmental issues, clean energy sources are in urgent demand not only to meet energy needs, but also to reduce carbon dioxide emissions. Having gained decades of operational experience and benefiting from ongoing development, nuclear power is clearly a good option for the fulfillment of future power requirements, because nuclear reactors do not emit the gases that cause the greenhouse effect. The next spell of nuclear power plant construction seems about to begin.

The cost of building a power reactor and obtaining the fission fuel is high. Operational optimisation, particularly in the use of the nuclear fuel, can result in considerable cost savings. For small-scale research reactors, the optimal use of fuel can also extend the life of the reactor, or improve the emission of neutrons required by scientific and medical research and applications.

The main concerns in nuclear fuel-management optimisation are the loading location of different fuel assemblies, burnable poisons and/or the control rods in the reactor core. The problem can be formulated as a combinatorial optimisation problem, which may be of a massive scale, and computationally challenging even with today's best computing facilities.

Tools are required to automate the optimisation process, and a considerable amount of work has been done in this field. However, advanced and more generic optimisation tools are still in urgent demand. This has driven the application of the Estimation of Distribution Algorithms (EDA), a class of algorithms with promising performance and supported by relatively sound theory, to the nuclear fuel-management optimisation problems.

The EDAs provide a promising optimisation tool. In addition, their flexibility enables them to be easily improved using problem-dependent information and/or parallel computing technology.

1.2 Overview

We present an overview of the thesis in this section, in which the Estimation of Distribution Algorithm is applied to the solution of nuclear reactor fuel-management optimisation problems.

Chapter 1: Introduction

An introduction to this thesis, outlining the project structure. The contents include the main drives of this project and an overview of the whole thesis.

Chapter 2: Nuclear Reactor Fuel Management Optimisation

We introduce basic knowledge of reactor fuel management, particularly the in-core fuel loading pattern optimisation problems. This allows the identification of generic questions for which computational algorithms can be designed.

Chapter 3: Review of Previous Work

In this chapter, previous work on reactor fuel management is presented. The 'white box' and 'black box' approaches are introduced and their advantages and disadvantages are discussed. The current state-of-the-art Genetic Algorithms (GAs) are introduced with more details to demonstrate their applications in this area.

Chapter 4: A Review of the Estimation of Distribution Algorithms

The Estimation of Distribution Algorithms (EDAs) are examined with the purpose of providing an advanced optimisation tool for this type of problem with a flexible and generic structure that enables the algorithm to be further improved with little effort.

Chapter 5: Estimation of Distribution Algorithms for the Travelling Salesman Problem

The Travelling Salesman Problem (TSP), a classical combinatorial optimisation problem, is chosen as an analytical problem to test the EDAs. A few EDAs are proposed and tested on a set of TSPs. The results are compared to some Genetic Algorithms with well-studied crossover operators, followed by a discussion of the EDAs' behaviour and performance when applied to TSPs.

Chapter 6: Estimation of Distribution Algorithms for Nuclear Reactor Fuel Management Optimisation

The objective of this chapter is to demonstrate how the EDAs can be adapted to real-world reactor fuel-management optimisation. The proposed EDAs are explained using a hypothetical problem.

Chapter 7: Applications and Results

Test cases are derived from the Imperial College CONSORT Reactor. The neces-

sary modelling, simulation and performance-enhancement work are explained. The experimental results of the EDAs are compared to benchmark GAs, followed by a discussion.

Chapter 8: Conclusions and Future Work

We summarize the project and some conclusions are drawn. Finally, we discuss possible future research directions.

Chapter 2

Nuclear Reactor Fuel Management Optimisation

2.1 Background

Nuclear Reactor Fuel Management Optimisation (NRFMO, Turinsky and Parks (1999) and Turinsky et al. (2005)) problems represent a range of optimal decision-making tasks, from the amount and physical properties of the fuel inventory, to the loading positions of different fuel elements and/or control rods, i.e. the Loading Pattern (LP). The problem of LP optimisation has attracted the most interest. In this work, we use 'fuel management optimisation' interchangeably with 'LP optimization,' unless otherwise specified. The general objective is to maximize reactor performance and/or minimize the cost, subject to operational and safety constraints.

There are various types of reactors currently in operation. The designs of the fission fuel are also very diversified in terms of their enrichment, size, shape, mass etc. Because of that, the fuel-management optimisation problem is very context-dependent.

In order to tackle the LP optimisation, it is both necessary and sufficient to understand the following two aspects:

(1) how the reactor works in principle, and how to load fission fuels;
(2) how to abstract the optimisation problem to a generic and well-understood form.

In this chapter, we will use a typical Pressurized Water Reactor (PWR), the VVER 440/230 (The Paks Nuclear Power Plant Company (2007)) reactor, as an example to illustrate the necessary information relating to LP optimisation. The reason for this is that the PWR is one of the most stable and popular reactor designs used for power plants in the last few decades. More importantly, the energy-generation process and fuel-loading operation of PWRs is very similar to most other reactors.

2.1.1 Understanding the Principles of a Reactor

The Reactor Power System

In the VVER reactor, fission-fuel assemblies develop chain reactions in the core vessel so that a massive amount of heat is generated inside the core. The coolant medium, pressurized water in this case, absorbs the heat and is directed into steam generators to produce high-pressure steam. The steam is then used to run electrical generators. Figure 2.1 is an illustrative picture of this power system.

In the energy-generation process, the amount of fission-fuel assemblies and their relative positions are crucial to the creation and maintenance of the chain reactions, which are the power source. The reactor's performance is therefore determined by the core structure and the in-core loading pattern.

The Reactor Fuel Loading

In this VVER example, a total of 349 fuel assemblies can be inserted into the reactor core. The cross-section of the fuel assemblies (fuel elements) is hexagonal. Each consists of a bundle of 126 smaller fuel rods. There are three types of fresh fuels, with enrichment levels of 1.6, 2.4 or 3.6% (all the rods in a given assembly are of equal enrichment). To ensure safety, there are 37 control rods, 30 of which are

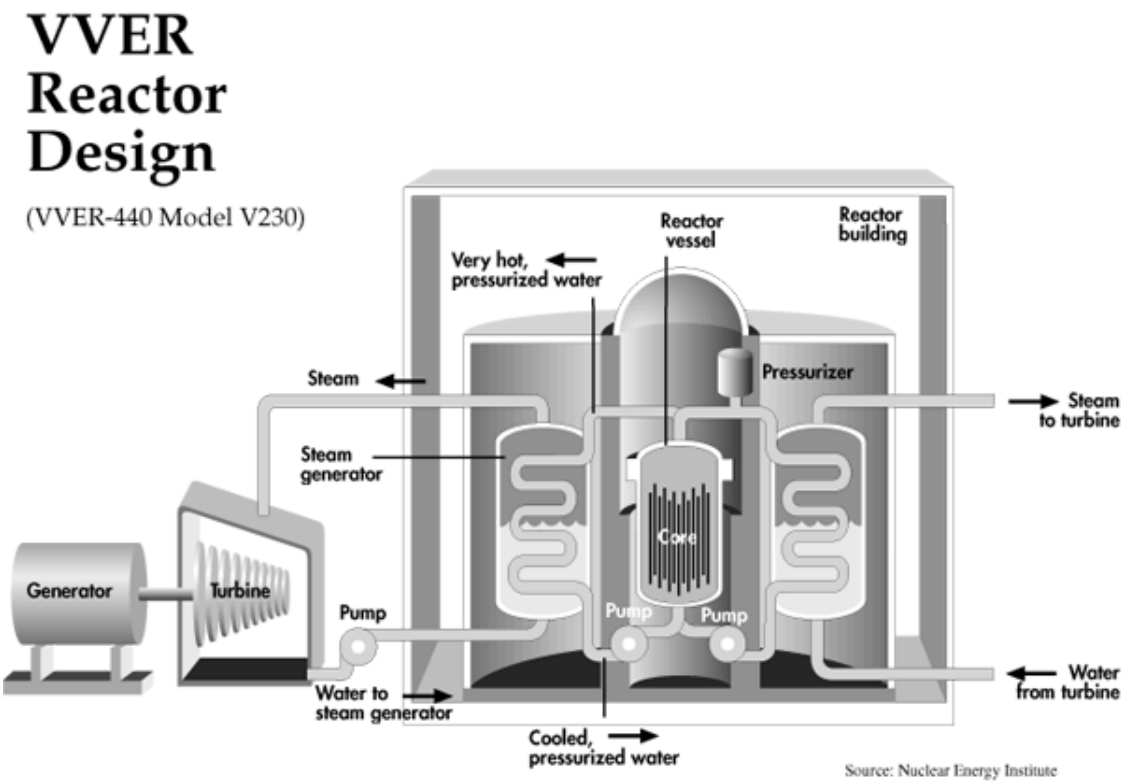


Figure 2.1: The VVER 440/230 reactor design: A typical pressurized water reactor. Inside the reactor vessel is the reactor core, where all stick-shaped fuel assemblies are.

out of core during operation, and seven rods are inserted. A top view of a loaded VVER 440 core is shown in figure 2.2.

There are two types of loading operation. The first one is initial loading. A full LP is required for the initial loading when the reactor starts, assigning 349 fuel assemblies to 349 positions, in this case.

The other one is the re-filling operation. To produce energy continuously, fresh fuels are required to be inserted into the reactor core, periodically replacing burnt ones. In most cases, instead of replacing the whole core, only part of the fuel assemblies are replaced with new fuel assemblies. After refilling, the in-core assemblies may also be shuffled for better performance. If long-term running is planned, a few re-loading operations may be needed. Therefore the problem is to find several successive LPs.

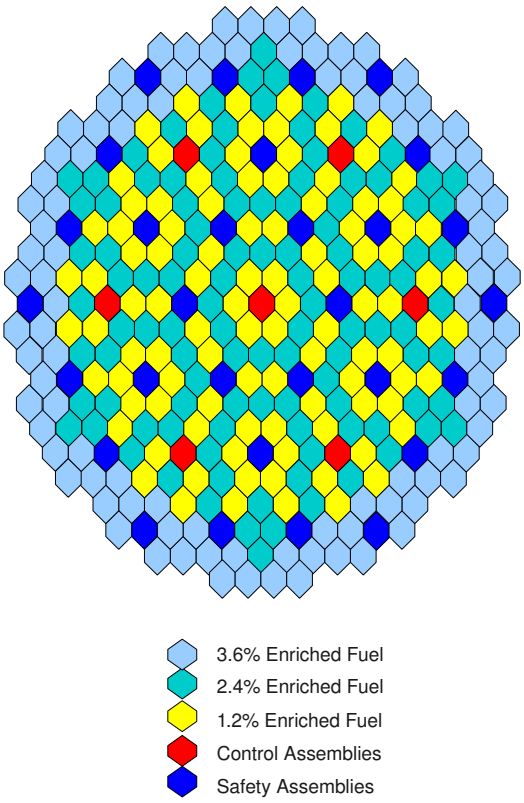


Figure 2.2: The Core Map of the VVER 440/230 : A Typical Pressurized Water Reactor. Each block represents a loadable position that might be occupied by different types of fuel assemblies or control rods. There 349 fuel channels and seven control rods in this LP.

For different reactor designs, the re-filling operation may be performed after the reactor is fully shut down, or when it is still online. In case of online re-filling, it may not be convenient to perform too many in-core shuffling operations.

2.1.2 Abstraction of the Optimisation Problem

Despite the complexity of nuclear reactors, most LP optimisation problems can be abstracted as an assignment problem, which is to assign the fuel elements to certain loadable positions in the reactor core with respect to a pre-defined objective function.

For either an initial loading or a re-filling operation, the LP optimisation problem is to use n items to fill m positions, where m is the number of in-core fuel channels and n is the number of fuel elements (the in-core ones plus the new fuels

replacing the old ones). In some previous work, the number of new fuel rods to be replaced is also referred to as the 'batch size' (Ziver et al. (2004)), which is equal to $(n - m)$.

For example, if we need to insert two fresh fuels into the VVER 440, replacing two highly burnt ones to keep up the chain reaction, then $n = 349 + 2 = 351$ and $m = 349$. In the case of initial loading, $n = m = 349$.

If multiple re-filling operations need to be optimised, the optimisation problem is regarded as a 'multi-cycle' LP optimisation. Compared to single-cycle LP, multi-cycle LP optimisation is a more difficult problem, due to its size, but there is no obvious difference in principle. We focus on the single-cycle problem, but the optimisation method developed will be capable of solving multi-cycle problems with little modification.

Most of the reactor LP optimisations can be abstracted as the n to m assignment problem, as described above. Based on this abstraction, generic methods can be developed to solve similar problems.

2.2 The Objective Function

In a commercial nuclear power plant the maximization of profit is the main goal to be achieved within a set of safety- and operationally-related principles. The objective functions are normally derived from some of the key parameters of the reactor. Examples include the Radial Form Factor (RFF, the ratio of the peak power of a fuel assembly to the average power of all in-core assemblies); the Average Discharge Irradiation (DI, the overall energy extracted from one fuel element); and 'Per Day Profit' (see Ziver et al. (2004)). These functions are independent of the type of reactor design.

In research reactors the objective is different, because they are not intended to produce power, but for scientific or medical experiments. Normally the objective is to maximize the amount and/or transport of the emitted neutrons (van Geemert et al. (1998)), or to extend the reactor's lifetime so that more experiments can

be carried out with a fixed fuel-element inventory (Franklin et al. (2005)). Both objectives are associated with the maximisation of the effective multiplication factor (K_{eff}) within certain constraints.

In an optimisation process, the objective function(s) will be evaluated many times for different candidate solutions. Since the necessary physical experiments are not feasible, the only way to do it is through a computer simulation. Adequate simulation software is able to provide accurate predictions of all of the key measurements of the reactor. However, even with the most up-to-date computing facilities, the reactor simulation is still very computationally expensive, which is a big obstacle for many optimisation techniques.

Furthermore, due to the complexity of the nuclear reaction and the reactor system, many pre-defined objective functions, as mentioned above, are non-linear and have multiple local minima. This increases the difficulties of the optimisation.

2.3 The Variables and the Search Space

Normally, the controllable variables are the amount and properties of the fuel elements and their positioning; the objectives can be a function of the reactor's key measures. The LP optimisation is difficult due to the massive possible search space and the complexity of the objective function.

The size of the problem space, i.e. the number of different LPs, depends on the number of fuel channels in the reactor core, m, and the number of fuel assemblies (the fuel inventory) available, n. In the PWR example, if there are 349 fuel assemblies to load the core, the number of possible LPs is 349!.

If we only consider the fuel types (according to enrichment level), there are only three different fuels in this PWR. If there are enough fuel assemblies for each type, the search space is $3^{349} \approx 3.28 \times 10^{166}$. It should also be noted that core symmetry is normally applied in LP design, which greatly reduces the problems' size. If octant symmetry is applied, the search space is approximately 10^{20}. It is still too large to perform exhaustive search, but various optimisation algorithms can

be used to achieve reasonable results.

Another example is one of the test cases used in this thesis, which is a 24-channel reactor with 5 different fuel types. The search space is 5^{24} (of the order of 10^{16}), which is similar to a typical PWR after applying the fuel-type categorizing and core symmetry.

Because of the size of the search space and the complexity of the objective function, it is impractical to use manual methods to identify optimal LPs. Thus an optimisation algorithm is required to automate the search.

2.4 The Constraints

The most important constraint in reactor fuel-management optimisation is safety. The generated power and power density need to be controlled within the mechanical design capacity. Some of the more frequently used core reactivity constraints are the shutdown margin and the power-peaking constraint (van Geemert et al. (1998)).

The reactivity constraint is a limitation on the absolute maximum reactivity of the core. To produce consistent power smoothly, the reactor needs to be slightly supercritical, so that the chain reaction can continue. If the core reactivity is over a certain limit, energy may be generated so fast that the coolant may not be able to transfer the power (and bring down the core temperature) quickly enough. The core structure may become unstable under very high temperatures and perhaps start melting.

The shutdown margin, as defined by the United States Nuclear Regulatory Commission, NRC (2008), is the instantaneous amount of reactivity by which the reactor is sub-critical or would be sub-critical from its present condition assuming all full-length control rod assemblies (shutdown and fine-control) are fully inserted, except for the single rod with the highest integral worth, which is assumed to be fully withdrawn. It is important that there is enough negative reactivity capable of being inserted by the control rods to ensure that complete shutdown is possible at all times during the cores lifetime.

The power-peaking factor can be regarded as the maximum power of all the fuel channels over the averaged power. It is normally expected to be minimized so that the power distribution in the core is flattened. Advanced computer modelling and simulation is able to predict these parameters accurately. Candidate LPs will be tested before being used.

Other types of constraints are operationally-related. For example, the availability of certain fuel elements, the structure and symmetry of the core, etc. In some cases, fuel shuffling may not be performed for the whole core, but only a few channels that are close to each other due to core structure and mechanical difficulties. Another example is that in many reactor designs the central positions of the core may always be occupied by certain fuel assemblies with relatively high reactivity.

The operational constraints will normally be handled as pre-defined rules and hard-coded into optimisation tools. Any candidate solution that breaks the rules will be screened out.

The timescale of the optimisation process can also be important, because a reactor may require relatively frequent reloading. Some reactors with the online reloading feature require the optimisation to be done in a few hours, or an even shorter time. An example is the CANada Deuterium Uranium (CANDU) reactor (Huo and Xie (2005)).

Chapter 3

Review of Previous Work

In this chapter we discuss previous work in reactor fuel loading-pattern optimisation. A considerable number of fuel-management optimization methods have been developed. They can be categorized into two groups: the 'white box' and the 'black box' approaches.

3.1 The 'White Box' Approaches

The 'white box' methods require a good understanding of the problem and of how the control variables are related to the optimisation targets. The objective function will need to describe the relationship between the control variables and the target parameters explicitly, ideally in linear/nonlinear equations, so that standard mathematical analysis can be used to direct the optimisation. Alternatively, background knowledge, constraints and mathematical modelling details are extracted as 'rules' to make decisions among various candidate solutions.

The so-called 'white box' methods applied to reactor fuel-management optimisation include gradient-based search (Ahn and Levine (1985), Suh and Levine (1990) and Balakrishnan and Kakodkar (1993)); linear programming (Sauar (1971) and Miller and Eckhoff (1975); dynamic programming (Wall and Fenech (1965)); mixed integer programming (Kim and Kim (1997), Mahlers (1997) and Quist et al. (1999)); and knowledge-based methods (Galperin and Kimhy (1991) and Lin et al.

(1998)).

In the above methods, a loading pattern X normally consists of a number of fuel-element variables, indicating the fuel elements loaded in the core. The index of the variables represents a loadable region or channel in the core. The actual values are some chosen physical property, depending on what the objective function is and how the objective function is related to it. It can be the enrichment of fissile material, the amount of burnable poison or the fuel burnup (Ahn and Levine (1985)) loaded at a fuel channel or, alternatively, a measure of reactivity, the K_∞, as in Melice (1969) and Sauar (1971).

For example, in Sauar (1971), a generic form of an LP can be described as:

$$X = [K_\infty^*] = [K_{\infty 1},\ K_{\infty 2},\ K_{\infty 3},\ K_{\infty i},\ ...,\ K_{\infty n}]^T \tag{3.1}$$

where X is a loading pattern and K_∞^* is the geometrical distribution of K_∞ in the reactor core; $K_{\infty i}$, $i = 1, 2, ...n$ can be regarded as the K_∞ value of the fuel element loaded at channel i without losing generality.

The objective is to minimise the cost subject to core reactivity and power level constraints. The problem is then modelled in a linear system:

$$min\, C^T \cdot X \tag{3.2}$$

subject to a few linear constraints, where C is a coefficient vector. A simpler method is used in Sauar (1971) but the implementation is not explained.

The key content of Melice (1969) and Sauar (1971)'s work is the use of K_∞ as the control variable, to formulate the linear system of the objective function and the constraints. The idea of representing the LP using a basic fuel property has been well recognized and used in much other work, including both conventional 'white box' methods and stochastic evolutionary algorithms.

Similarly, in Ahn and Levine (1985), part of the objective function is maxi-

mizing the K_{eff} of a PWR. The LP is represented by enrichment and burnup:

$$max\, OF = max\, K_{eff} = max\, f(X) = max\, f(e, BU) \tag{3.3}$$

where e is the geometrical distribution of enrichment in the core and BU is the burnup distribution. Considering the burnup as a fixed value, it can be shown that:

$$OF = f(e, BU) = f(e) \tag{3.4}$$

A steepest-descent search (Snyman (2005)) can be applied to maximise K_{eff}:

$$e_{t+1} = e_t + \gamma \cdot \nabla K_{eff} \tag{3.5}$$

$$\nabla K_{eff} = \frac{dK_{eff}}{de_t} = [\frac{\partial K_{eff}}{\partial e_t^1},\ \frac{\partial K_{eff}}{\partial e_t^2},\ ...\frac{\partial K_{eff}}{\partial e_t^i},\ ...\ \frac{\partial K_{eff}}{\partial e_t^n}]^T \tag{3.6}$$

where t is the iteration number; $\frac{dK_{eff}}{de_t}$ is the gradient of K_{eff} to the enrichment distribution e at iteration t; $\frac{\partial K_{eff}}{\partial e_t^i}$ is the partial derivative of K_{eff} with respect to the i^{th} fuel channel and γ is a scalar. This is a simple form of the gradient-descent search method. The loading pattern X is modified gradually in the direction in which the objective function, K_{eff}, increases most. The resulting solution is the optimal distribution of enrichment.

The gradient of K_{eff}, $\frac{\partial K_{eff}}{\partial e_t^i}$ is calculated by a separate simulation code. Implementation details are not given in Ahn and Levine (1985). Such a calculation depends on the reactor diffusion model. A discussion of different models is beyond the scope of this thesis. However, if an explicit model is defined, then such a gradient can be calculated with reasonable effort.

When considering the constraints, a gradient-projection method can be used instead of the original steepest-descent method. The gradient of K_{eff} is projected on to the feasible-solution region before being used as the direction of the next step for searching.

$$e_{t+1} = PR(e_t + \gamma \cdot \nabla K_{eff}) \tag{3.7}$$

where PR is the projection of enrichment distribution e_{t+1} onto the feasible region. The projection can be computed by setting:

$$PR(e_{t+1}) = mid\{e_{t+1}, {}^{l}e, {}^{u}e\} \tag{3.8}$$

where $mid\{\cdot\}$ is the median of the given set of values, and ${}^{l}e$ and ${}^{u}e$ are lower and upper bounds of the possible enrichment level, e.g. 1% and 3%.

Instead of using a fuel channel-wise or geometrical region-wise array to represent an LP, in Balakrishnan and Kakodkar (1993), the authors use another approach to represent the core loading. They have used five variables:

$$X = [x_1, x_2, x_3, x_4, x_5], \tag{3.9}$$

to represent the number of fuel elements to be loaded; their averaged relative distances from sensitive areas (a_1 and a_2) in terms of reactivity variations, the centre of the core, and the nearest core periphery, respectively.

The objective function to be minimised is:

$$OF = (W_b B)^2 + (W_c C)^2 + (W_t T)^2 + (W_1 S_1)^2 + (W_2 S_2)^2 \tag{3.10}$$

where B is the maximum bundle power; C is the maximum channel power; T is the maximum coolant channel outlet temperature; S_1 and S_2 are the decreases in the two sensitive areas a_1 and a_2; and Ws are a set of weights for different decision objectives.

Since all the decision objectives are functions of an LP represented by the five chosen variables x_1 to x_5, the objective function can be written as:

$$OF = Y^T \cdot Y \tag{3.11}$$

where

$$Y = \begin{vmatrix} W_b B(x_1, x_2, x_3, x_4, x_5) \\ W_c C(x_1, x_2, x_3, x_4, x_5) \\ W_t T(x_1, x_2, x_3, x_4, x_5) \\ W_1 S_1(x_1, x_2, x_3, x_4, x_5) \\ W_2 S_2(x_1, x_2, x_3, x_4, x_5) \end{vmatrix} \tag{3.12}$$

The Jacobian matrix of the objective function is:

$$J = \begin{vmatrix} W_b \frac{\partial B}{\partial x_1} & W_b \frac{\partial B}{\partial x_2} & ... & W_b \frac{\partial B}{\partial x_5} \\ W_c \frac{\partial C}{\partial x_1} & W_c \frac{\partial C}{\partial x_2} & ... & W_c \frac{\partial C}{\partial x_5} \\ W_t \frac{\partial T}{\partial x_1} & W_t \frac{\partial T}{\partial x_2} & ... & W_t \frac{\partial T}{\partial x_5} \\ W_1 \frac{\partial S_1}{\partial x_1} & W_1 \frac{\partial S_1}{\partial x_2} & ... & W_1 \frac{\partial S_1}{\partial x_5} \\ W_2 \frac{\partial S_2}{\partial x_1} & W_2 \frac{\partial S_2}{\partial x_2} & ... & W_2 \frac{\partial S_2}{\partial x_5} \end{vmatrix} \tag{3.13}$$

The Gauss method is then used to perform the minimisation, given the variables X and the Jacobian J:

$$J^T \cdot J \cdot \Delta X = -J^T \cdot Y \tag{3.14}$$

where ΔX decides the search step and direction, i.e.

$$X_{t+1} = X_t + \Delta X; \tag{3.15}$$

Iterating this process, the objective function will decrease steadily until it is converged. The key issue is the calculation of J consisting of the partial derivatives, e.g. $\frac{\partial B}{\partial x_1}$. Unlike the direct steepest-descent method and gradient projection method discussed above, which used a core simulation model-dependent code to compute the gradient, an estimated gradient is used here. The gradients are estimated using:

$$\frac{\Delta B}{\Delta x_1}, \frac{\Delta B}{\Delta x_2}, ..., \frac{\Delta C}{\Delta x_1}, ..., \frac{\Delta T}{\Delta x_1}, ... \tag{3.16}$$

where Δx_is are small arbitrary changes made to the control variables and ΔB, ΔC etc. are the corresponding changes in objective function caused by Δx_i.

It should be noted that the variable chosen for this method needs to be mapped back to a position-wise core configuration during the optimisation to form an LP. The implementation details are not given, but it is clear that the optimised X cannot be mapped to a unique core configuration directly. Users may need a randomized routine to do it automatically.

A possible problem for all of the numerical methods discussed is that the resulting solution, either a K_∞ reactivity distribution or enrichment distribution, may not be feasible, even if the constraints have been satisfied. This is because of the set-up of the problem, and/or the optimisation algorithm used, which allow variables to change continuously. For instance, the calculated optimal enrichment may be 2.5% at channel j, but it is possible that only 1.5%, 2% and 3% enrichment levels are available. Users may need to round the real values to obtain a practical solution in such a situation. This will affect the accuracy of the solution.

A non-algorithmic loading pattern search method was proposed (Galperin and Kimhy (1991) and Lin et al. (1998)), which uses a 'rule-based' expert system. Starting from a reference loading pattern, new candidates are proposed by modifying the initial one slightly, under the control of the pre-defined rules. The rules are defined by experts with extensive knowledge of the particular reactor and will improve the designated objective function and/or the satisfaction of the constraints.

Example rules include: the fuel elements with the highest reactivity should be loaded at the core centre at all times (Lin et al. (1998)); slightly burnt fuel assemblies may not be adjacent to each other if they are in the central area of the core (Galperin and Kimhy (1991)), etc. The rule sets are highly problem-dependent.

The mechanism of the 'white box' methods consists of two key steps. Firstly, a model explicitly describing the optimisation problem is built; secondly, optimisation algorithms using the information extracted from the problem descriptions are applied; they are normally gradients of the objective function or 'if - then - else'

rules.

It should be noted that once the model is built, any suitable algorithm can be applied, including those that do not use much of the modelling information.

Problem-dependent information benefits 'white box' approaches by allowing a faster search of a local area, and it may also reduce the number of expensive reactor-core calculations. The disadvantages are that these methods require too much problem-dependent or even simulation code-dependent information, which is not always available. Furthermore, there is no mechanism for these methods to escape from a local optimum, and they cannot be readily generalised. These methods often employ a simplified physics model rather than a full physics model in order to construct less complicated mathematical equations to perform optimisation, e.g. performing a fast gradient calculation (Ahn and Levine (1985), Smuc et al. (1994)). There will be some search bias due to the simplifications incorporated in these methods.

3.2 The 'Black Box' Approaches

The 'black box' methods do not use knowledge of how the objectives are related to the control variables, but only ask for an evaluation of the candidate solutions, from whatever methods are available. So the explicit form of the objective function is hidden in a 'black box'.

Typical examples are Simulated Annealing (SA, Parks (1987), Parks (1990)), Smuc et al. (1994), Stevens et al. (1995), Kropaczek and Turinsky (1991), Kropaczek et al. (1991) and etc); Genetic Algorithms (GAs, Poon and Parks (1993), Parks (1996), Chapot et al. (1999), Erdogan and Geckinli (2003), Pereira and Lapa (2003), Ziver et al. (2004) and etc); Artificial Neural Networks (ANNs, Sadighi et al. (2002), Faria and Pereira (2003) and Ortiz and Requena (2004)); Tabu Search (TS, Lin et al. (1998), del Campo and Francois (2002) and Castillo et al. (2004)); and Ant Colony Optimisation (ACO, Machado and Schirru (2002)).

The 'black box' normally stands for a simulation software package or any

form of computational routine that is able to evaluate a given LP. The simplest optimisation process is a two-step iteration: the 'black box' is first asked to evaluate the current solution, and the second step is to propose new candidates based on the feedback.

The SA method is one of the most successful 'black box' algorithms. In Kropaczek and Turinsky (1991), an LP is represented in binary form:

$$X_{l,m,n,o} = \begin{cases} 1 & \text{Fuel assembly of type } m, \\ & \text{burnable poison loading } n, \\ & \text{and orientation } o \text{ in location } l \\ 0 & \text{Otherwise} \end{cases} \tag{3.17}$$

A few objective functions are used. Without losing generality, we refer to $f(X)$ as the objective function to be minimized. $f(X)$ is computed by a fast and reliable neutronics model. The SA method starts with an initial guess of an LP X_0 with its $f(x)$ calculated by the neutronics model. A new candidate LP is generated by making small changes to the previous LP. The new LP will be accepted if the objective function decreases. Otherwise, a probability of acceptance is calculated using an exponential function:

$$P_{t+1} = e^{\frac{f_t - f_{t+1}}{c}} \tag{3.18}$$

where P_{t+1} is the probability of accepting the LP at step $t+1$; c is a scalar parameter called the temperature. The initial temperature is normally set to a large value. c is decreased during the search using the following equation:

$$c_{t+1} = \omega \cdot c_t \tag{3.19}$$

where ω is a user-defined scalar indicating the 'cooling' rate. Normally it is set in $[0, 1]$, e.g. 0.9. This is to ensure that, at the beginning, any LP is likely to be acceptable, even it increases the objective function, but gradually, as the temperature

cools down, SA turns from random exploration to a greedy search and only LPs decreasing the objective function are accepted. Therefore the SA method is more capable of overcoming local optima compared to conventional methods.

Similarly, a SA is used in Parks (1990) combined with performance enhancements, including the incorporation of experience and knowledge to filter out poor LPs; a modified cooling method to allow more random search even at the end of the search; and natural evolution-inspired parallel runs.

In Lin et al. (1998), del Campo and Francois (2002), the TS method is used as a deterministic algorithm that keeps a 'tabu' memory so that the recently evaluated LPs will not be generated and tested again. In TS, an initial LP is chosen and small random changes are made to it, but only those that improve the objective function and have not been evaluated recently (in a user-defined period of preview search, e.g. the last three iterations) are accepted. The visiting history is saved in a separate data structure and the search goes on to the next iteration, until a stop criterion is met. This method can reduce the redundant computational cost considerably.

Many TS applications are also combined with problem-dependent information to screen out LPs that are believed not to be acceptable without being evaluated by the full neutronics model. For example, in Hmaida et al. (1999) certain perturbations, such as swapping two fuel elements with large reactivity differences, are not allowed.

One of the most important features of the 'black box' methods is that they clearly separate the reactor-model simulation and the optimisation process. Such methods facilitate the application of generic or newly developed optimisation technologies in LP optimisation without extensive knowledge of nuclear engineering.

The 'black box' algorithms can be generalised because no problem-dependent information is needed. They have already shown some advantages in resolving problems with complicated and poorly understood search spaces compared to conventional methods (de Jong and Spears (1989)). However, problem-dependent information can always be used when available to enhance the performance.

The main drawback of this type of method is that it needs a large number of objective function evaluations, normally done by running a reactor neutronics simulation model, which can be extremely computationally expensive.

3.3 The Current State of The Art: the Genetic Algorithm

A considerable amount of work on applying 'black box' optimisation methods to the reactor fuel-management optimisation problem has been carried out in the past few decades. Among these methods, the current state of the art is the GA. It has attracted interest in recent years. The GA's performance on various reactors loading-pattern optimisation problems is very promising (Poon and Parks (1993), Parks (1996), Chapot et al. (1999), Erdogan and Geckinli (2003), Pereira and Lapa (2003) and Ziver et al. (2004)). Therefore GAs are used as the benchmark in this work.

Inspired by Darwin's evolutionary theory, the GA was first presented by Holland in the 1970s (Holland (1975)). Another comprehensive book on the GAs' concepts and applications is by Goldberg (Goldberg (1989b)). Other general overviews and introductions are also available, such as Beasley et al. (1993) and Back et al. (1997).

After decades of development, the modern GAs are capable of handling various types of data structures used for encoding anything other than the binary string (Michalewicz (1997) and Back et al. (1997)), e.g., real vectors, permutation vectors, matrix etc. Many different types of crossover and mutation operators have been developed to deal with the representation data structures, e.g. the Cycle Crossover and the swap mutation for permutation representation (Oliver et al. (1987)).

3.3.1 The GA_HTBX for Reactor Fuel Management

Poon (1992) and Poon and Parks (1993) is the earliest research on the application

of the GA with Heuristic Tie Breaking Crossover (GA_HTBX) to reactor LP optimisation. A considerable of amount of research has since been carried out on this subject. A generic GA framework for the loading-pattern optimisation can be summarized as follows:

(1) initialize a population of candidate loading patterns;
(2) compute the objective function (fitness) values for all the candidates;
(3) select promising candidates according to their fitness value;
(4) perform crossover and mutation on the selected individuals, to propose new solutions;
(5) evaluate the current candidate pool, and stop if certain criteria are met, e.g. the maximum number of LP evaluations. Otherwise go back to step (3).

The key considerations of applying GAs are the representation of the problem, the crossover and mutation operator, and the setting and tuning of the parameters used to adjust the algorithm, such as the population size, the crossover and mutation rate, and the maximum number of function evaluations.

For the reactor loading patterns, a natural way of encoding the LP is to use a permutation vector (Poon (1992)), which represents the ordering of items (fuel elements) to given locations (fuel channels). The crossover/mutation operators designed for permutation vectors can be employed to solve LP optimisation. Studies on the use of crossover operators for ordering problems have been done, such as in Oliver et al. (1987) and Poon and Carter (1995). The main concerns are how to keep the permutation vector valid and improve the optimisation results.

Based on the generic study, the GA_HTBX is especially designed to solve the reactor LP optimisation encoded in a permutation vector with 2D structure. It uses fuel elements' reactivity (K_∞) ranking numbers as their IDs. An LP is then represented by a 2D arrangement of the in-core fuel IDs, which geometrically maps the reactor core structure. In other words, the LP is represented in a 2D ranked

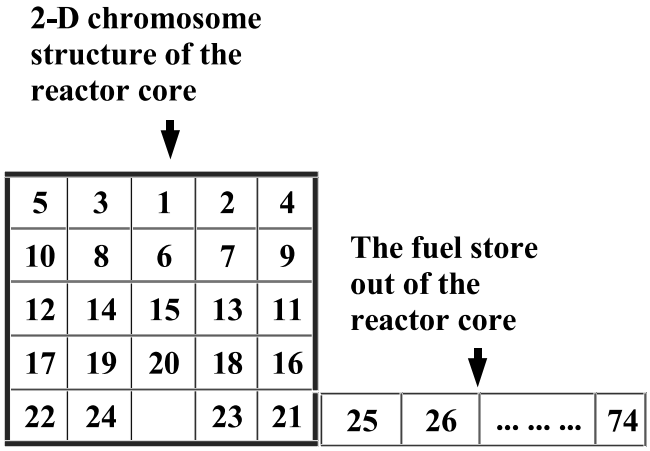


Figure 3.1: An illustration of an LP in 2D form used in the benchmark GA_HTBX. The first 24 integer numbers are the IDs of the fuel channels, and the rest of them are IDs for the out-of-core storage positions.

K_∞ distribution.

For example, the structure of an LP of a reactor with 24 in-core fuel channels and 74 available fuel elements can be illustrated as in figure 3.1. The permutation encoding must include all the available fuel elements, whether they are in-core or out-of-core, so that the full search space can be explored. In this example, there are 74 fuel assemblies in total, so the LP representation (or 'chromosome' in GA terminology) is a permutation from 1 to 74.

Since only the in-core LP affects the reactor directly, only the first 24 positions (in-core fuel channel) have a 2D structure. The rest of them, position 25 to 74, are used to represent the available fuel elements that are not loaded in this loading pattern.

The workflow of HTBX described in Poon (1992) is to rank the fuel elements by their reactivity and transform two loading patterns into permutations, then choose a 'chunk' of fuel assemblies that are adjacent, from two candidate LPs, and swap these two chunks. The resulting two offspring LPs are not necessarily valid permutations. A tie-breaking algorithm is applied to them to generate final LPs. See figure 3.2 for details of HTBX.

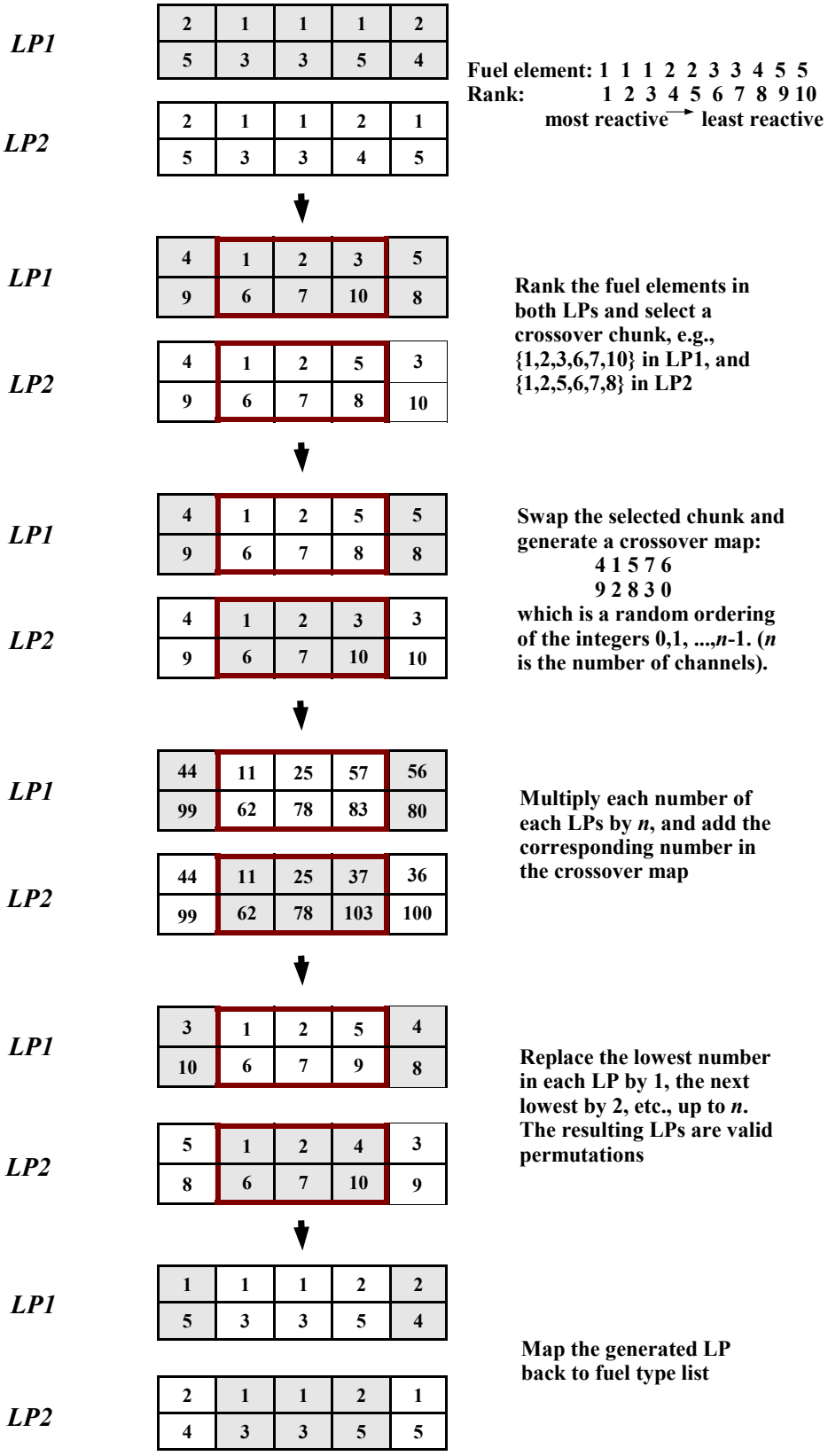


Figure 3.2: An example of HTBX in GA for an ordering problem

It should be noted that when HTBX swaps the sub-LPs, the swapping part must be chosen from the in-core part. In this example, it is between 1 and 24. The reason for this is that swapping the out-of-core part will not change the reactor status, and therefore make HTBX inefficient.

GAs can be sensitive to control parameters, such as the population size and the choice of crossover operator. Research regarding the parameter control in GAs was established in de Jong (1975). The use of GA_HTBX's parameters in reactor-fuel LP optimisation is studied in Poon (1992).

A hybrid GA with expert knowledge has been successfully applied to PWR LP optimisation (DeChaine and Feltus (1996)), in which the authors conclude that swapping physically adjacent sub-LPs (similar to HTBX) and biasing the search to known good areas can improve the GA's performance. Multi-objective GAs (Parks (1997) and Toshinsky et al. (2000)) have demonstrated the capability of the GA to solve multi-objective problems, and have provided a good tool for examining the trade-off between different objectives of a complex problem.

Additionally, parallel GAs technologies have been proposed and successfully applied to a reactor's water-feeding system optimisation in Pereira and Lapa (2003). A few independent GA processes located on different PCs with periodic communication are set up (the so-called 'island' model). The authors found that not only is the computational time reduced by utilizing multiple networked personal computers, but the optimisation results are also improved. The reason may be the better randomization. Although it is not exactly a fuel-management problem, the approach can be readily applied to LP optimisation problems. In Zhao et al. (1998), a similar GA is employed for generic reactor LP optimisation.

The main drawback of the GAs' application is that it normally requires a large number of objective function evaluations before converging on satisfactory results. For those applications having very computationally expensive objective functions, such as the reactor LP optimisation, GAs may not be able to provide reasonable solutions within reasonable time scales. The other issue is that despite the enormous

effort, evolutionary algorithms, including the GA, still lack an adequate theory that describes their properties.

3.4 Summary

In this chapter, we reviewed the previous optimisation research in reactor fuel management, which is categorized into two groups. The current state-of-the-art, GA, is then briefly reviewed, with a discussion of its application in reactor fuel-management optimisation.

Based on this review, and considering the main difficulties in reactor fuel-loading pattern optimisation, three possible approaches can be proposed for further improvements in efficiency and accuracy:

(1) the development and use of more advanced optimisation algorithms;
(2) the construction of fast surrogate models (e.g. using Artificial Neural Networks) to mimic reactor simulation codes;
(3) and/or the application of parallel computing techniques.

In this thesis, we focus on developing new optimisation algorithms to solve the LP optimisation problems. The GA is used as a benchmark for comparison study.

Chapter 4

A Review of the Estimation of Distribution Algorithms

4.1 Background

The development of heuristic search and optimisation methods is driven by two reasons (Pearl (1984)). Firstly, many hard optimisation problems from real-world applications and academic research are difficult to solve in a reasonable time scale, because the search space is massive and/or the objective function is very computationally expensive. Therefore exhaustive search is not practicable. Secondly, generic and widely-applicable optimisation algorithms are attractive to engineers because they can be easily customised for different problems with little effort and are still able to provide near-optimal results.

According to their behaviour, two types of heuristic search strategies are available. One of them always produces the same result under the same conditions, and is known as deterministic methods; the other type is non-deterministic methods. To overcome possible local minima, random perturbations provide a non-deterministic element that may produce a better exploration in the search space, and results in uncertainty of the outcome for the same initial conditions. The non-deterministic methods are also known as stochastic algorithms. The best-known members of this

kind are Simulated Annealing (SA) and Genetic Algorithms (GA, Holland (1975) and Goldberg (1989a)).

Some of the stochastic algorithms maintain multiple candidate solutions instead of keeping only one solution in each iteration. They simulate the natural evolution process in the pool of candidates by selecting the ones with better objective function values, and generating new solutions by combining the features from these ones. These methods are collectively categorised as Evolutionary Algorithms (EAs) or Evolutionary Computation (EC). The best-known subset of EAs is GAs. Others include Evolutionary Strategies (ES, Rechenberg (1973)), Genetic Programming (GP, Koza (1992)) and Differential Evolution (DE, Price et al. (2005)).

The Estimation of Distribution Algorithms (EDAs, Mühlenbein and Paab (1996) and Larrañaga and Lozano (2001)) are relatively new algorithms in the EAs' catalogue. The EDAs are an extension of conventional GAs aiming to improve their capacity for solving not-naturally-encoded problems in which the crossover may perform poorly. They also estimate possible correlations between the input variables (Müehlenbein and Hons (2004)).

In EDAs, the candidate solutions are generated by sampling a probability distribution model. The model usually starts with a uniform distribution and is gradually updated by the information extracted from promising solutions selected from the candidate pool. The rest of the algorithm is very similar to the GAs.

In GAs, the area being searched and the variable dependency are hidden in the encoded solution and the crossover operator. Using a probability distribution model in EDAs, the area containing promising solutions is represented explicitly. The variable dependencies can also be incorporated into the model.

The EDAs share most of the GAs' advantages, such as generality and ability to overcome some local minima. Furthermore, by perturbing the explicit model of the promising search area with knowledge and experience, problem-dependent information can be easily incorporated.

4.2 The Basics of the EDAs

In this section, we describe the outline of a basic EDA and demonstrate the basic EDA idea through a simple example. The main steps of an EDA are:

1. initialise the probability model to a uniform distribution or pre-defined distribution;
2. sample new solutions from the probability model and calculate their objective function values;
3. select some individuals from the current population, based on their objective function values;
4. revise the probability model using the information extracted from selected individuals;
5. if stop criteria are not met, go to step 2, otherwise end the search.

The key considerations are how to construct, use and update the probability models.

To better understand EDAs, we will consider the Univariate Marginal Distribution Algorithm (UMDA, Mühlenbein and Paab (1996)), and use it to solve a onemax problem. The optimisation problem is:

$$\max f(X) = \sum_{i=1}^{3} x_i = x_1 + x_2 + x_3; \tag{4.1}$$

where the input variable X is:

$$X = [x_1, x_2, x_3], \qquad x_i \in \{0,\ 1\},\ \ i \in \{1, 2, 3\} \tag{4.2}$$

The basic UMDA approach to this problem is illustrated step by step.

1. **Initialise a uniform distribution model** P. We use ${}^{t}P$ to represent the distribution model at iteration t, here $t = 0$. The data structure of the probability model is a real-valued vector P having the same dimension as X, in which each p_i represent the probability of x_i being 1.

$$ {}^{0}P = [p_1, p_2, p_3] \ = [0.5, 0.5, 0.5] \tag{4.3} $$

2. **Sample** M **individuals (**$M = 6$ **for this case) from the model** ${}^{0}P$**, and compute their objective function values**. The sampling method is like tossing a coin. Firstly, initialise an empty individual ${}^{t}_{j}X$, where t is the generation number and j is the individual ID in the current population. Here $t = 0$ and $j = 1$. For each p_i, generate a uniformly distributed random number r_i between $[0, 1]$:

$$ \begin{aligned} r = \quad & [r_1, r_2, r_3] \quad = \quad [0.4, \quad 0.5, \quad 0.1] \\ {}^{0}P = \quad & [p_1, p_2, p_3] \quad = \quad [0.5, \quad 0.5, \quad 0.5] \\ {}^{0}_{1}X = \quad & [x_1, x_2, x_3] \quad = \quad [-, \quad -, \quad -\,] \end{aligned} \tag{4.4} $$

Compare each r_i and p_i, if $r_i < p_i$, then set $x_i = 1$, otherwise, $x_i = 0$. Since $0.4 < 0.5$, $0.5 = 0.5$ and $0.1 < 0.5$:

$$ {}^{0}_{1}X = [1, 0, 1] \tag{4.5} $$

and the objective function is calculated:

$$ f({}^{0}_{1}X) = 1 + 0 + 1 = 2 \tag{4.6} $$

Similarly, other individuals can be generated. The current population is:

$$\begin{aligned}
{}^{0}_{1}X &= [1,0,1] \quad f({}^{0}_{1}X) = 2 \\
{}^{0}_{2}X &= [1,1,1] \quad f({}^{0}_{2}X) = 3 \\
{}^{0}_{3}X &= [1,0,0] \quad f({}^{0}_{3}X) = 1 \\
{}^{0}_{4}X &= [0,1,0] \quad f({}^{0}_{4}X) = 1 \\
{}^{0}_{5}X &= [1,0,1] \quad f({}^{0}_{5}X) = 2 \\
{}^{0}_{6}X &= [0,0,1] \quad f({}^{0}_{6}X) = 1
\end{aligned} \tag{4.7}$$

3. **Select the N best individuals with the highest $f(X)$, e.g.** $N = \frac{M}{2} = 3$. The selected IDs are $j \in \{1,2,5\}$:

$$\begin{aligned}
{}^{0}_{1}X &= [1,0,1] \\
{}^{0}_{2}X &= [1,1,1] \\
{}^{0}_{5}X &= [1,0,1]
\end{aligned} \tag{4.8}$$

4. **Update the probability model by counting the frequency for each x_i in the selected individuals** The updating method is described in the following equation:

$$^{1}P(x_i) =^{1} P(x_i = 1) = \frac{\sum_j x_{ij}}{N} \tag{4.9}$$

in which the updated ^{1}P will be the sampling model at generation/iteration 1; i is the ID number of the input variables; j is the ID of the selected individuals and N is the total number of the selected individuals.

Hence

$$
\begin{aligned}
{}^1P(x_1 = 1) &= \frac{\sum_j x_{1j}}{N} = \frac{\sum_j x_{1j}}{3} = 1 \\
{}^1P(x_2 = 1) &= \frac{\sum_j x_{2j}}{N} = \frac{\sum_j x_{2j}}{3} = \frac{1}{3} \\
{}^1P(x_3 = 1) &= \frac{\sum_j x_{3j}}{N} = \frac{\sum_j x_{3j}}{3} = 1
\end{aligned}
\tag{4.10}
$$

The updated P is:

$$
{}^1P = [1, \frac{1}{3}, 1] \approx [1.0, 0.3, 1.0] \tag{4.11}
$$

5. **Stop if maximum number of generations is reached, otherwise go back to 2 and sample new solutions**. The current best solution is:

$$
\begin{aligned}
{}^0_2X &= [1, 1, 1] \\
f({}^0_2X) &= 3
\end{aligned}
\tag{4.12}
$$

If the maximum number of generations is set to 2 (starting from 0), then the algorithm will repeat the above process once and terminate. In the next generation, the sampling model is:

$$
{}^1P = [1.0, 0.3, 1.0] \tag{4.13}
$$

${}^1_1X$ can be sampled by comparing a set of random numbers r_1, r_2 and r_3 to ${}^1P$, if $r_i < p_i$, then set $x_i = 1$, otherwise, $x_i = 0$,

$$
\begin{array}{rcccc}
r & = & [r_1, r_2, r_3] & = & [0.6, \quad 0.7, \quad 0.3] \\
{}^1P & = & [p_1, p_2, p_3] & = & [1.0, \quad 0.3, \quad 1.0] \\
{}^1_1X & = & [x_1, x_2, x_3] & = & [1, \quad 0, \quad 1]
\end{array}
\tag{4.14}
$$

and

$$
f({}^1_1X) = 1 + 0 + 1 = 2 \tag{4.15}
$$

Similarly, other individuals can be sampled:

$$
\begin{aligned}
{}^1_1X &= [1,0,1] \quad f({}^1_1X) = 2 \\
{}^1_2X &= [1,1,1] \quad f({}^1_2X) = 3 \\
{}^1_3X &= [1,0,1] \quad f({}^1_3X) = 2 \\
{}^1_4X &= [1,1,1] \quad f({}^1_4X) = 3 \\
{}^1_5X &= [1,0,1] \quad f({}^1_5X) = 2 \\
{}^1_6X &= [1,0,1] \quad f({}^1_6X) = 2
\end{aligned}
\tag{4.16}
$$

Select the N best individuals with the highest $f(X)$ ($N = 3$), the selected ID $j \in \{2, 4, 1\}$:

$$
\begin{aligned}
{}^1_2X &= [1,1,1] \\
{}^1_4X &= [1,1,1] \\
{}^1_1X &= [1,0,1]
\end{aligned}
\tag{4.17}
$$

If the objective function values are equal, solutions are selected randomly.

Update the probability model by counting the frequency for each x_i being 1 using the following equation:

$$
{}^2P(x_i) =^2 P(x_i = 1) = \frac{\sum_j x_{ij}}{N} \tag{4.18}
$$

Hence

$$
\begin{aligned}
{}^2P(x_1 = 1) &= \frac{\sum_j x_{1j}}{N} = \frac{\sum_j x_{1j}}{3} = 1 \\
{}^2P(x_2 = 1) &= \frac{\sum_j x_{2j}}{N} = \frac{\sum_j x_{2j}}{3} = \frac{2}{3} \\
{}^2P(x_3 = 1) &= \frac{\sum_j x_{3j}}{N} = \frac{\sum_j x_{3j}}{3} = 1
\end{aligned}
\tag{4.19}
$$

The updated P is:

$$
{}^2P = [1, \frac{2}{3}, 1] \approx [1.0, 0.7, 1.0] \tag{4.20}
$$

Since the maximum number of generations, 2, is reached, the algorithm is terminated. The best solutions found by this search process are:

$$\begin{aligned} {}^{0}_{2}X \;&=\; {}^{1}_{2}X \;=\; {}^{1}_{4}X \;= [1,1,1] \\ f([1,1,1]) &= 1+1+1=3 \end{aligned} \tag{4.21}$$

The UMDA is one of the simplest forms of EDA. However, its idea is the basis of all variants. It estimates the probability of the distribution of the promising solutions and maintains it with a separate data structure, in which the input variables are considered as independent of each other.

In this basic version, the distribution model is re-estimated in each generation. There are other variants of UMDAs using slightly different updating methods. For example, in the original UMDA paper, Mühlenbein and Paab (1996), an alternative implementation is introduced, which uses an incremental learning strategy to update the distribution model:

$${}^{t+1}P(x_i) = (1-\lambda)\cdot {}^{t}P(x_i) + \lambda \cdot \frac{\sum_{j=1}^{N} x_i}{N} \tag{4.22}$$

in which λ is a scalar, and N is the number of selected candidates.

Through this example we demonstrate how an UMDA performs an optimisation in order to understand a basic EDA. The key element is the use of the probability model.

4.3 Members of the EDA Family

EDAs can be divided into two different classes, according to the input variable dependencies considered. The first class assumes there are no dependencies between the variables of a candidate solution. The second class of EDAs takes into account these variable dependencies. The probability distribution models for the variables with dependencies are slightly more complicated than the one with no variable dependency.

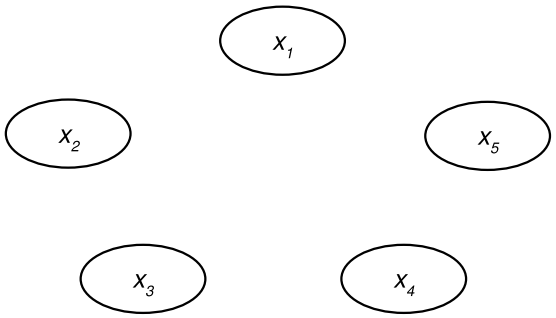


Figure 4.1: An illustration of the non-variable dependencies model used in the EDAs. (e.g. UMDA and PBIL)

4.3.1 Without Variable Dependencies

The best known non-dependency EDAs are the UMDAs introduced in the last section, Population-Based Incremental Learning (PBIL, Baluja (1994)), and Compact Genetic Algorithms (cGA, Harik et al. (1999)). In a candidate solution $X = [x_1, x_2, x_3, x_4, x_5]$, the relationship between each of the components can be graphically represented in figure 4.1.

Because of the independence, the probability distribution of promising solutions $P(X)$ in any EDA without variable dependency is:

$$P(X) = [p_1, p_2, ...p_n] = [p(x_1), p(x_2), ...p(x_n)] \tag{4.23}$$

where $p(x_i)$ is the appearance frequency of x_1 being '1' in the selected promising solutions (in the case of binary encoding without loss of generality). New solutions are sampled from this model by sampling x_i from each $p(x_i)$ separately.

Updating the probability model is the key difference between the basic PBIL and UMDA. Given a binary vector encoding, e.g. a candidate solution $X = [0, 1, 0]$, in PBIL, the probability model is updated with the best individual in the generation:

$${}^{t+1}P(x_i) = (1 - \lambda) \cdot {}^{t}P(x_i) + \lambda \cdot {}^{t}P(x_i^*) \tag{4.24}$$

in which λ is a scalar, also known as the 'learning rate;' $P(x_i^*)$ is the probability of x_i being 1 in the best solution found in the current generation, X^*. $P(x_i^*)$ can only be 1 or 0. It can be observed that if the top N best solutions are used, the basic PBIL is the same as the UMDA described in equation 4.22. In Fyfe (1999), the application of PBIL to dynamic problems and multi-objective problems is studied.

Another non-dependency EDA is the Compact Genetic Algorithms (cGA, Harik et al. (1999)). The probability vector P is the same as the UMDA and PBIL. Usually the initial p_is are set to 0.5. The population of cGA only contains two individuals competing with each other. The winner X^* will be used to update the probability model. For p_i and x_i in X:

$$^{t+1}P(x_i) = {}^tP(x_i) + \frac{1}{2} \cdot {}^tP(x_i^*) \tag{4.25}$$

The algorithm terminates when P has converged, i.e. all $p_i = 1$.

Since there is no variable dependency in such EDAs, the probability distribution is estimated from the current generation of candidate solutions and combined with the previous distribution. The main difference between these variants is the method used to combine the previous distribution with the newly estimated one.

4.3.2 With Variable Dependencies

In terms of the topology, the possible dependency structures include the bivariate and multivariate dependencies. In bivariate dependencies, each variable depends on one and only one other variable. Multivariate dependency allows one variable to depend on multiple variables.

Bivariate Dependency

The bivariate dependency structure can be categorised into pairwise, tree and forest structures. The pairwise dependency means each of the variables is interacting with one and only one other variable, as shown in figure 4.2.a. Tree dependencies mean

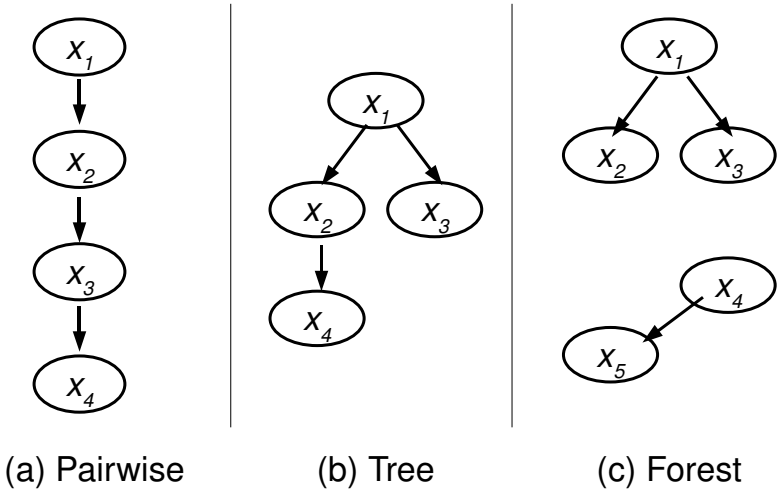


Figure 4.2: An illustration of the probability model with bivariate dependencies. (a) shows the pairwise dependency; (b) shows the tree structure and (c) shows the forest structure of dependencies

that each variable may depend on one and only one other variable, but more than one variable can be dependent on the same variable; see figure 4.2.b. If a few such trees exist, the dependency is regarded as a forest, as in figure 4.2.c.

An example of an EDA with pairwise dependencies is Mutual Information Maximisation for Input Clustering (MIMIC, Bonet et al. (1996)). The main difference between MIMIC and the EDAs without variable dependency such as UMDA and PBIL is the construction of the probability model.

In MIMIC, it is assumed that a 'linked list' type of variable dependency, as shown in figure 4.2.a, represents the distribution of the promising solutions. The probability distribution model is:

$$P_\pi(X) = [p(x_{i_1} \mid x_{i_2}), p(x_{i_2} \mid x_{i_3}), \cdots, p(x_{i_{n-1}} \mid x_{i_n}), p(x_{i_n})] \tag{4.26}$$

where $\pi = i_1, i_2, ..., i_n$ is a permutation of variable ordering; $p(x_{i_{j-1}} \mid x_{i_j})$ is the probability of $x_{i_{j-1}}$ given x_{i_j}. The conditional probability can be calculated by using the appearance frequencies, $p(x_{i_{j-1}})$, $p(x_{i_j})$ and $p(x_{i_{j-1}} \wedge x_{i_j})$ in the selected

solutions:

$$p(x_{i_{j-1}} \mid x_{i_j}) = \frac{p(x_{i_{j-1}} \wedge x_{i_j})}{p(x_{i_j})} \tag{4.27}$$

The key consideration is to decide the optimal permutation π. In MIMIC, it is learnt from the selected solutions. The estimated distribution $P_\pi(X)$ from the selected individuals should have maximal agreement with the true distribution $P(X)$. The Kullback-Leibler divergence is used to measure this agreement, which can be derived as a function of

$$h_\pi(X) = h(x_{i_n}) + \sum_{j=1}^{n-1} h(x_{i_{j-1}} \mid x_{i_j}) \tag{4.28}$$

where $h(x_{i_j})$ is the entropy (Shannon (1948)) of the variable x_{i_j}, and $h(x_{i_{j-1}} \mid x_{i_j})$ represents the mutual information between the interacting variables $x_{i_{j-1}}$ and x_{i_j}. The optimal permutation π is needed to minimise $h_\pi(X)$.

There are $n!$ possible permutations. For computational efficiency, a greedy search is used in MIMIC to search for the optimal permutation. Firstly, minimum i_n is located:

$$i_n = \text{argmin}_j h(x_j) \tag{4.29}$$

The minimum $h(x_j \mid x_{i_n})$ is then found and the process is repeated. It can be described as:

$$i_k = \text{argmin}_j h(x_j \mid x_{i_{k+1}}) \tag{4.30}$$

where $j \neq i_{k+1}...i_n$ and $k = n-1, n-2, ..., 2, 1$.

Having the optimal variable dependency permutation π, the conditional probability distribution can be calculated as in equation 4.26. New individuals can be sampled from P_π using the same sampling method used in UMDA and PBIL.

The probability model is rebuilt with every iteration, since the optimal variable dependency permutation may change. There is no direct incremental relationship between ${}^{t+1}P$ and ${}^{t}P$.

The Bivariate Marginal Distribution Algorithm (BMDA, Pelikan and Mühlenbein

(1999)) is based on a similar idea, but the probability model used is an acyclic dependency graph, or a set of trees (forest). For example, as in figure 4.2.c, the distribution model P is:

$$P(X) = [p(x_1), p(x_2 \mid x_1), p(x_3 \mid x_1), p(x_4), p(x_5 \mid x_4)] \tag{4.31}$$

The dependency structure in BMDA is learnt from the selected individuals. The χ^2 statistics (Mitchell (1997)) are used in BMDA to measure dependencies between variables. If the dependency is larger than a threshold value, then these two variables are considered as dependent.

Using the χ^2 dependency, a greedy search is applied to find a dependency tree or forest of the variables, in which the dependency between the variables is a maximum. Firstly, an arbitrary variable is chosen and the dependencies between it and all previously chosen variables are calculated. A variable pair is considered as dependent if it has the maximum χ^2 dependency. The process is repeated until all the variables are processed.

Once the variable dependency structure is established, the estimated distribution P as in equation 4.31 can be calculated. New individuals can be sampled from it using the same method used in UMDA and PBIL. The distribution model P is rebuilt in every generation, since its structure may change as the search goes on.

It can be observed that in EDAs that consider the variable dependency, such as MIMIC and BMDA, the dependency structure of the probability model needs to be learnt online using a measurement of dependency-entropy in MIMIC and χ^2 in BMDA. In BMDA, the dependency structure learnt might be one tree or a forest, while in MIMIC the structure is always a linked list.

Compared to a non-dependency model, learning the probability model structure is an extra component added to a basic EDA. Once the model structure is established, the estimation of distribution and sampling are the same as with a simple EDA, such as the UMDA.

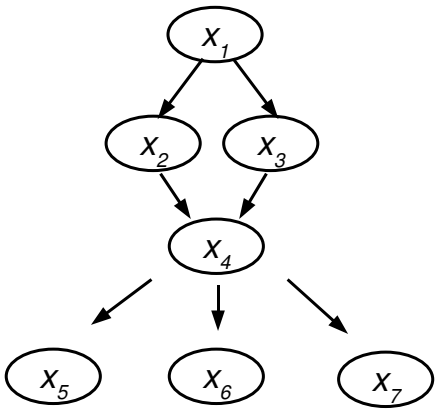


Figure 4.3: An illustration of the probability model with multivariate dependencies

Multivariate Dependency

In a multivariate dependency structure, a variable may depend on more than one other variable. This dependency structure is like an acyclic graph. Figure 4.3 illustrates a multivariate structure.

The Factorised Distribution Algorithm (FDA, Mühlenbein et al. (1999)) was introduced to solve an additively decomposable problem. The distribution model P in FDA as in figure 4.3 is:

$$P(X) = [p(x_1), p(x_2 \mid x_1), p(x_3 \mid x_1), p(x_4 \mid x_2 \wedge x_3), p(x_5 \mid x_4), p(x_6 \mid x_4), p(x_7 \mid x_4)] \tag{4.32}$$

Unlike the BMDA or MIMIC, the dependency structure is known beforehand. Therefore in an FDA there is no structure learning. Users need to calculate the conditional probabilities shown in equation 4.32 in every generation.

The Estimation of Bayesian Networks Algorithm (EBNA, Etxeberria and Larrañaga (1999)) uses a Bayesian network to capture the joint probability distribution from the selected promising solutions. A local search method is used to search for the near-optimal network structure, and the Bayesian Information Criterion to

evaluate the quality of a candidate network. The probability model is learnt online in every generation.

The Bayesian Optimisation Algorithm (BOA, Pelikan and Mühlenbein (1999)) uses a very similar idea to EBNA. The difference lies in its use of a Bayesian Dirichlet equivalent to measure the quality of a dependency network structure, and of a greedy search to find the near-optimal network.

Ideally, the variable dependencies used in the EDAs is known beforehand, so the probability model is structural-ready and does not need to be re-learnt in every generation. However, learning the variable dependency from samples is a difficult problem in its own right, and can be very hard and computationally intensive. (Etxeberria and Larrañaga (1999)).

4.4 Applications of the EDAs

As an extension of the original GAs, EDAs can be applied widely across many engineering, academic and business areas. For example, Zhang et al. (2004) use hybrid EDAs to solve a set of classical continuous optimisation problems. In Zhang et al. (2005), an EDA with a newly proposed Guided Mutation is applied to maximum clique problems. In Li and Zhang (2004), another hybrid EDA for multi-objective knapsack problems is developed and Jiang et al. (2006) apply the EDAs with heuristic information to the reactor fuel loading-pattern optimisation. Since EDA applications are becoming popular, it is not possible to cover the full range of the current established works. Instead, we describe how the EDAs can be applied to typical optimisation problems by explaining the data structures commonly used in the various situations so that they can be utilised directly in similar applications.

To apply the EDA to optimisation problems, the problem input variables should be encoded first, so that the algorithm can build a distribution model for it. Similarly to the GAs, popular representation data structures include binary strings, integers and real vectors. Most of the problems can be regarded as either discrete or continuous problems. It should be noted that we will focus on the use of the EDAs

without variable dependency.

4.4.1 For Discrete Problems

For discrete problems, the EDAs can be easily applied, especially when using binary-string encoding (recall the UMDA example in the previous section). Similarly, integer ID encodings can also be handled if they are transformed to binary format. For example:

$$X = [x_1, x_2, x_3] = [1, 2, 3] \tag{4.33}$$

can be represented into binary strings easily:

$$X = [00, 01, 11] \tag{4.34}$$

The corresponding data structure of the probability model is:

$$P = [p_{11}, p_{12}, p_{21}, p_{22}, p_{31}, p_{32}] \tag{4.35}$$

Alternatively, another encoding is to use a binary matrix:

$$X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.36}$$

where each row has one and only one 1, the corresponding column ID (column 1, 2 and 3) is the actual integer value. The permutation vector is a special case of a normal integer vector, so similar methods can be used when dealing with permutation vectors. The corresponding P has the same data structure. A uniform distribution is written as below:

$$P = \begin{bmatrix} 0.33 & 0.33 & 0.33 \\ 0.33 & 0.33 & 0.33 \\ 0.33 & 0.33 & 0.33 \end{bmatrix} \tag{4.37}$$

If the variable dependency is used, the data structure of P needs to be modified slightly.

4.4.2 For Continuous Problems

While originally developed for discrete problems, the EDAs were adapted for continuous problems soon afterwards. An example is in Tsutsui et al. (2001), where the histogram model is used for continuous domain decomposition. By doing this, the EDAs treat a continuous domain as a finite number of sub-domains.

For a candidate solution X in a continuous search space (a, b) :

$$X = [x_1, x_2, x_3],\ x_i \in [a, b[\tag{4.38}$$

A histogram model is formed by dividing $[a, b]$ into S sub-domains, which can be equally spaced. The frequency of x_i falling in each sub-domain j, $Q(i, j)$ is counted on M selected candidates Xs, and the probability is written as:

$$P(x_i, j) = \frac{Q(i, j)}{M} \tag{4.39}$$

which represents the likelihood of x_i being in sub-domain j. Let $a = 1$, $b = 3$, $M = 5$, $S = 2$ - therefore the sub-domains are $[1, 2)$ and $[2, 3)$. Given the following selected promising Xs:

$$\begin{aligned} X_a &= [1, 2, 3] \\ X_b &= [1, 2, 2] \\ X_c &= [1, 1, 2] \\ X_d &= [3, 2, 3] \\ X_e &= [2, 2, 1] \end{aligned} \tag{4.40}$$

Then the probability of x_1 falling in the first sub-domain is

$$P(x_1, 1) = \frac{3}{5} = 0.6 \tag{4.41}$$

Similarly, the whole P is a 3×2 matrix. Each row represents a variable and each column represents a sub-domain.

$$P = \begin{bmatrix} 0.6 & 0.4 \\ 0.2 & 0.8 \\ 0.2 & 0.8 \end{bmatrix} \tag{4.42}$$

Alternatively, many EDAs for continuous optimisation make use of the normal distribution. Each variable is described by a normal density function, with two key parameters μ and σ, or mean and standard deviation respectively.

$$P(x_i) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{\frac{1}{2}\left(\frac{x_i - \mu_i}{\sigma_i}\right)^2} \tag{4.43}$$

For an input variable vector $X = [x_1, x_2, ...x_n]$, the data structure of P is identical to X but it has two associated real number vectors μ and σ.

P is updated by updating the μ and σ. In Rudlof and Koppen (1996), the Hebbian rule (Mitchell (1997)) is used.

$$\begin{aligned} \mu_{t+1} &= \mu_t + \alpha \cdot (b_t - \mu_t); \\ \sigma_{t+1} &= \sigma_t \cdot \beta \end{aligned} \tag{4.44}$$

where t is the generation number, b_t is the baricenter of the selected solutions at generation t, and β is a scalar. After the updating, new solutions can be sampled from the new normal distributions.

4.5 Discussion

4.5.1 On the Convergence

Because of the stochastic attributes of EDA, which are inherited from the original GAs, a proof for the convergence is difficult to construct. However, the presence of the probability model has motivated research in the EDAs' convergence theory. We introduce the available theoretical research results in this section.

In Mühlenbein (1998), Gonzalez et al. (2000) and Zhang (2004a), the authors conclude that, theoretically, the EDAs with the independent variable model are not capable of solving those problems with highly-interacting input variables. In Mühlenbein and Mahnig (1999), the authors investigate a class of FDAs applied to additively decomposable functions, such as:

$$f(x_1, x_2, x_3) = f_1(x_1) + f_2(x_2) + f_3(x_3) \tag{4.45}$$

The results imply that the tested FDA converges to the global optimum.

The research work in Zhang (2004b) suggests that using a higher-order variable dependency could improve the EDA's chance of converging to the global optimum, and the UMDA may be trapped at every local minimum. In Zhang (2004a) and Zhang and Müehlenbein (2004), the authors discuss and prove the convergence of the FDA with proportional and truncation selection on additively decomposable problems with variable overlapping, i.e.

$$f(x_1, x_2, x_3) = f_1(x_1, x_2) + f_2(x_2, x_3) \tag{4.46}$$

with an infinite population size.

More general conclusions are drawn that if accurate variable dependency is used, it is sufficient for an EDA to converge to the global optimum for additively decomposable problems. The proof is based on the infinite population size.

4.5.2 On the Complexity

In Gao and Culberson (2005), the space complexity of the class of EDAs with variable dependencies is proved as exponential in the problem size, even if the actual variable dependency is not complicated and loose. If an independent variable model is used, the complexity is linear.

Since the construction of the probability model and the sampling operator in the EDAs may vary according to the choice of probability model, the time complexity can vary significantly in practice. For the independent variable model, such as in the UMDA, the time complexity varies linearly with problem size. For the EDAs with variable dependency, another optimisation problem of finding the best dependency structure greatly increases the overall computational cost. If the dependency between one variable and all other variables is considered, such as:

$$P(x_i) = P(x_i \mid x_1, x_2, ...x_{i-1}, x_{i+1}, ...x_n) \tag{4.47}$$

where P is the probability model, x_i is one of the input variables, and n is the problem size, the time complexity of measuring dependencies and finding the optimal structure varies exponentially with problem size.

In practice, the complexity of EDA algorithms can be controlled by simplifying the dependencies dimension (such as MIMIC) and using local search methods to find a near-optimal dependency structure. The trade-off between the algorithm's complexity and its performance needs to be decided carefully according to the requirement and available resources.

4.6 Summary

In this chapter, we have reviewed the Estimation of Distribution Algorithms (EDAs), a set of relatively new evolutionary algorithms extended from the original Genetic Algorithms (GAs). The basic idea and implementation of a simple EDA is described through an example. The classification of different EDAs is introduced briefly, fol-

lowed by a discussion of how to apply the EDAs to practical problems.

In EDAs, a separate distribution probability model is employed from which the new solution is sampled, instead of using a crossover operator. This feature enables the EDAs to handle problems for which the GAs' crossover may fail to perform, as might happen, for example, when the input variable is not naturally encoded. In the meantime, the probability model may consider the dependency between variables, so that the distribution of promising solutions can be more accurately represented.

A considerable amount of theoretical research on the EDAs has been done. The convergence of the EDAs with accurate variable dependency on additively decomposable problems have been proved for an infinite population size.

The EDAs are more computationally expensive than the conventional GAs, because of the probability model. If an advanced probability model is used, the search and construction of the variable dependency structure is a hard optimisation problem in its own right, and the complexity varies exponentially with problem size. In some situations, this may not be suitable for practical applications.

The EDAs are based on a critical assumption that the distribution of promising solutions can be estimated by a finite number of samples. This may affect the accuracy of the EDAs.

Chapter 5

Estimation of Distribution Algorithms for the Travelling Salesman Problem

Having reviewed the basics of the EDAs, it is essential to understand how they work in practice. In this chapter, we use the Travelling Salesman Problem (TSP), a special case of the Quadratic Assignment Problem (QAP), as the test problem, in order to understand the EDAs when applied to real-world problems, to study practical performance-enhancement methods and, finally, to compare them with benchmark algorithms.

The reasons for which we choose TSP are three-fold. Firstly, as a member of the black-box type of method, the EDAs are designed to be easily applied to a wide range of problems with little effort. Hence the EDAs developed for TSPs should be nearly ready for more complicated applications, such as the reactor fuel loading pattern optimisation. Secondly, TSPs are classical combinatorial optimisation problems. Plenty of well-established research work has been done, including the application of the GAs to TSPs. Some of the test problems and GAs are borrowed from recognisable research studies, including Poon (1992) and Poon and Carter (1995), and re-implemented for comparison studies.

Finally, the similarity between TSPs and the reactor loading pattern optimisation problems is obvious and well-recognised by many researchers. The two problems can both be described as assigning items to locations. This encouraged Chapot et al. (1999) and Poon (1992) to use the TSP as an analytical test problem before applying their algorithms to the real-world reactor loading pattern optimisation problems.

As pointed out by Goldberg (Goldberg (1989a)), some GAs perform well on some problems but very poorly on others, even though these problems are 'similar'. Although the TSP and the reactor loading pattern optimisation problems have obvious similarities in terms of the format of input variables (the 'items' and the 'locations'), there is no direct relationship between an algorithm's performance on a TSP and a reactor application. This is because different problems have different objective functions. A similar input variable space does not have anything to do with the shape of the objective function space. So no matter how smoothly and systematically an algorithm searches in one objective space, it can become easily stuck in another objective space.

This situation applies to the TSPs and the reactor loading pattern optimisation problems. Hence the reasons for using the TSPs as the analytical problems are: to provide a better understanding of how the EDAs handle the 'item'-'location' type of input variables; to provide a comparison with the benchmark algorithm; and to provide a test bed for researching some generic performance enhancement techniques that are not restricted to a particular problem.

5.1 Overview

In this chapter, we implement and apply the EDAs to a set of TSP problems and compare their performances to the benchmark GAs and a standard SA. We also suggest some problem-independent techniques that can be used in the EDAs to enhance their performance.

We describe the EDAs for TSPs first. The main considerations are the encoding method of TSPs, the type of probability model and how to maintain it, and the

candidate solution-generating method using a probability model sampling operator. We also suggest some performance enhancements based on the previous research and experience in using evolutionary algorithms to solve real problems.

The performance of the EDAs and the benchmark GAs and SA are compared on a set of TSPs. Some of the TSPs are taken from Poon (1992) and Poon and Carter (1995) because they are relatively simple and well understood. By carrying out the comparison, a better understanding of the EDAs can be achieved, as well as gaining experience in selecting proper algorithms and techniques for other applications.

5.2 Applying the Estimation of Distribution Algorithms to TSPs

As described in the last chapter, the algorithm structure of the EDAs is very similar to that of the GAs. They both search by evolving a population of candidate solutions. The main difference is the way in which they reproduce and maintain the candidate pool. The key points of the application of EDAs to TSPs (and other problems) are the encoding method for representing the candidate solutions, the type of the probability model, and how to use it to generate new solutions.

We discuss the encoding method used in the EDAs for TSP first, followed by the use of the probability model and by a discussion and study of some performance-enhancement methods.

5.2.1 Encoding for the TSPs

The TSPs are a set of classical combinatorial optimisation problems that look for the shortest route by which a salesman can visit a set of cities. Alternatively, it can be described as ordering a set of cities in such a way as to minimise the length of the path along which they lie. A TSP is easily abstracted as a permutation of integers, in which each integer represents a city. The permutation representation is one of the most natural ways of encoding TSPs, and has been used in the research

of GAs and SA for TSP.

There are other methods of encoding a TSP and they were mainly introduced in GA research. For example, the Ordinal Representation (Grefenstette et al. (1985)), the Adjacency Representation (Grefenstette et al. (1985)), the Adjacency Listing Representation (Whitley et al. (1989)), the Position Listing Representation (Poon (1992)) and the Precedence Matrix Representation (Fox and McMahon (1990)).

As pointed out by previous studies in GAs, the candidate solutions need to be encoded naturally into binary strings to produce good results (de Jong and Spears (1989)). In recent years, other type of data structures have been used in GAs, like the ones in the previous paragraph. These representations are normally associated with specially-designed crossover and mutation operators to produce valid and reasonable solutions.

However, it is generally believed that the GAs' performance on the TSP type of problems has not been particularly impressive (Poon and Carter (1995)). It is mainly because the way in which solutions are represented and reproduced in GAs (crossover and mutation) is not straightforward or 'natural'. The candidate solutions in the GAs contain the encoding of the problem explicitly, and the dynamics in the search process implicitly. Some hidden information, or the 'Schemata' in GA terminology, can easily be disrupted during the crossover and/or mutation process. The search is therefore expected to contain more random jumps and the search space may not be explored smoothly. In addition, a specially-designed representation and crossover operator may restrict the generality of black-box algorithms.

In the EDAs, we intend to use a natural encoding of the solutions, such as the permutation representation, to preserve the generality of the algorithm. At the same time, a separate distribution probability model is used to keep the statistical information gathered during the search or any relevant information. New solutions are generated by sampling the model. This mechanism separates the encoding of the candidate solution and the place of preserving the dynamics of the optimisation process (in the probability model).

It is clear that in such EDAs, the problem encoding is not associated with solution-reproducing operators. This enables users to use the most straightforward encoding method without looking for the associated crossover operator, or having to develop a new one. The EDAs' generality is also well preserved.

For the EDAs applied to TSPs, we use the permutation representation. A candidate solution of a TSP is represented in an integer vector that contains a permutation of integers from 1 to N, where N is the number of cities. A candidate tour of a five-city TSP is illustrated below:

$$\begin{aligned} \text{Cities} &: \{1, 2, 3, 4, 5\} \\ \text{Candidate Tour} &: \{3, 2, 1, 5, 4\} \end{aligned} \tag{5.1}$$

where the candidate tour represents a possible shortest path, visiting city 3 first, then city 2, city 1, city 5, city 4 and back to city 3.

Alternatively, this permutation can also be transformed to a binary matrix form, as shown in figure 5.1. The rows stand for the visiting order and the columns are the city IDs. Entry $[m, n]$ set to '1' means the n_{th} city will be visited at the m_{th} stop of the tour. Since each city should be visited once and only once, each row and column has one and only one '1' bit.

This binary matrix form of a permutation makes no difference in representing a tour, except it uses more storage. The reason for which we introduce the binary matrix is that it makes it easier to understand the relationship between the candidate tour and the probability model in subsequent sections. Hence the binary matrix representation is used more frequently in this work, despite the fact that in the actual computer programs it is always transformed to an integer vector, for efficiency reasons.

Despite the slight inefficiency in computational cost, the binary matrix helps understand the decomposition of the TSPs in the EDAs. A candidate tour of the TSP is a combination of some entries of the whole matrix, as illustrated in figure 5.1. These entries can be regarded as the building block of the TSPs. The binary

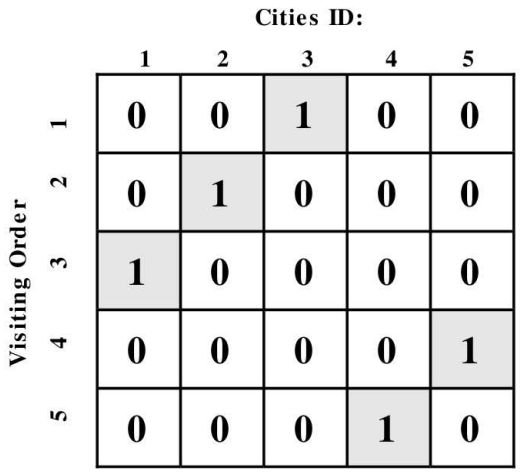


Figure 5.1: The binary matrix form of a Permutation Representation of a tour of a five-city TSP. Each row and each column only has one '1' bit

matrix contains all of the possible building blocks to form any candidate solution. The optimisation then consists of picking some of the matrix entries under certain rules.

5.2.2 Probability Model for Generating Candidate Solutions

The EDAs can be categorised into two sub-classes. One uses a relatively simple probability model that has no dependence between any input variable. The other one uses a probability model that does have input variable dependencies. The first one is obviously simpler and more computationally efficient, while the latter is generally believed to have better modelling capacity; however, it can be very complicated and computationally expensive.

The probability model in the EDAs is utilised to capture and preserve the dynamic information found during the search, as well as any relevant heuristic information. Given a complicated real-world application, the input variables - namely, the encoded candidate solutions - may or may not interact with each other. The dependent models are better at capturing these relationships, and ideally can improve

the search. The dependency structure can be decided beforehand, using background knowledge, or achieved on the fly during the search.

Despite the theoretical advantage of the probability models with variable dependency, relatively simple EDAs with a non-dependent model are frequently used in real-world applications (Zhang et al. (2004)). One reason for this is that the computational cost of the dependent models can be prohibitively high. One of the initial drives for using an EDA (or conventional GAs) was to find near-optimal solutions with reasonable time and effort (Poon (1992)). Therefore the application of the dependent models might not be cost-efficient.

The other reason is that, when used in the EDAs, the statistical data available for finding the dependencies is limited. The current candidate solutions are literally all the data that is available (the size of the current population), which may be insufficient for extensive statistical analysis, and may contain redundant individuals. We believe that the dependency models may fail to capture the variables' dependency accurately, due to incomplete data.

Finally, the performance of the EDAs with non-dependent models can be improved by combining with local search or heuristic information. Promising results have been well recognised. For example, in Zhang et al. (2004), an EDA with an independent probability model combined with derivative-free local search has been tested on a set of difficult continuous functions.

Because of the above reasons, we will apply an EDA with an independent probability model to the analytical TSPs. It is simple to understand and sufficiently well-analysed to be extended to the solution of very complicated real-world applications.

A Non-Dependent Model for TSPs

The best-known EDAs with the non-dependent distribution model are the UMDA - Univariate Marginal Distribution Algorithm (Müehlenbein (1998)), the PBIL - Population Based Incremental Learning (Baluja (1994)), and the cGA - Compact

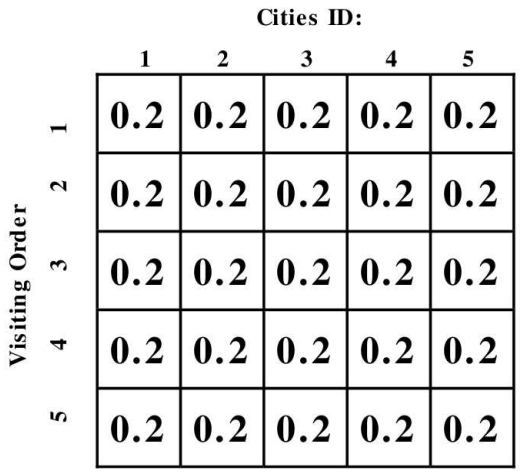


Figure 5.2: The data structure of the initial non-dependent probability model P for a hypothetical five-city TSP. Each entry represents the initial probability of visiting a city (column) at a certain stop (row)

Genetic Algorithm (Harik et al. (1999)).

For a five-city TSP, the data structure of the probability model used in this chapter is illustrated in figure 5.2. A matrix entry (m, n) represents the probability of visiting city n at the m_{th} stop. Initially, the model can be a uniform distribution if no background information is given. Hence each matrix entry is set to 0.2 because there are five cities to be chosen randomly.

The probability model updating method is similar to that for PBIL. It can be described as:

$$P_{t+1} = (1 - \alpha) \cdot P_t + \alpha \cdot X \tag{5.2}$$

where P is the probability model; t is the number of the iteration or generation; P^0 is a uniform distribution model; α is a scalar between $[0, 1]$; and X is the frequency of occurrence of each city-stop arrangement, extracted from the selected candidate tours. The independent model implies that the arrangements of cities to stops are independent of each other. An example is illustrated in figure 5.3.

Having updated the probability model by estimating the distribution of the

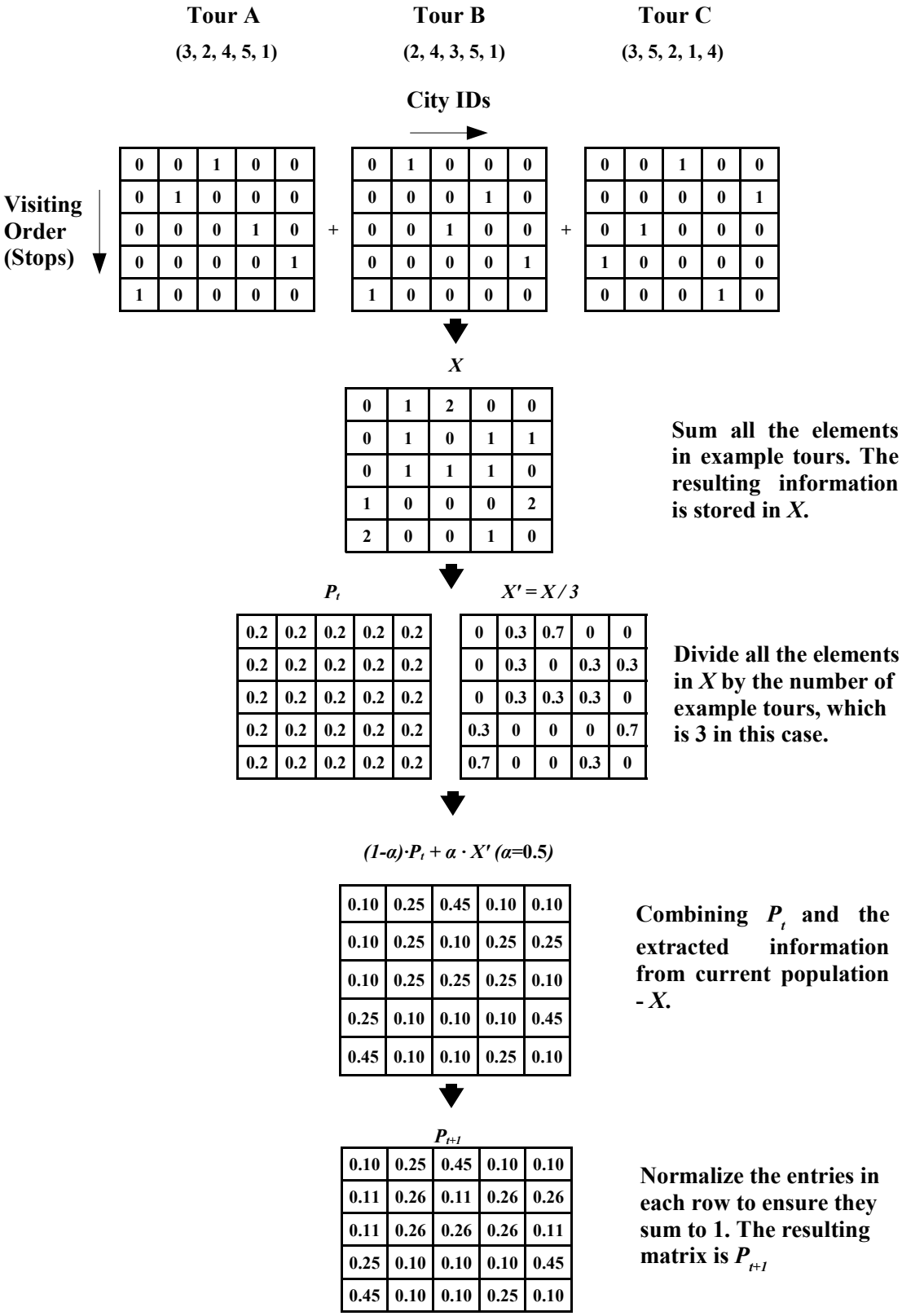


Figure 5.3: An example of updating a probability model in EDAs

promising solutions, new candidate solutions will be generated by sampling the model. An example of generating new tours for TSPs is illustrated in figure 5.4.

5.2.3 Use of Elitism, Local Search and Heuristic Information

To develop a more useful algorithm, ready to be applied to real-world applications with little effort, the standard optimisation algorithm must be improved with performance-enhancement methods. We discuss some practical techniques that can be used in the EDAs.

In real-world applications, EAs like GAs and EDAs are often combined with problem-dependent information and/or local search to improve their performance. Although many of the methods are highly problem-dependent and specially designed for very specific applications, some of the performance-enhancement methods are generalisable and have proved to be successful. Examples include the elitism strategy in the GAs and EDAs (Poon (1992) and Poon and Carter (1995)); the use of local search in various classical optimisation problems (Zhang et al. (2004) and Zhang et al. (2006)); the use of heuristic information in the GAs for reactor fuel loading pattern optimisation (Poon (1992) and Poon and Carter (1995)); and the Ant Colony Optimisation for TSPs (Dorigo and Gambardella (1997)).

Elitism Strategy

The elitism strategy is a frequently-used method in evolutionary algorithms like the GAs and EDAs. A common implementation is to keep the current best solution and ensure it is passed on to the next generation of the population (Poon and Carter (1995)). Clearly, the advantage of always preserving the current best is to ensure it will not be lost. In the EDAs, the same method can be used because the EDAs also maintain a population of candidate solutions.

In the GAs, the elitism strategy aims to keep the promising sub-pattern of building-blocks in the current best. The current best needs to be in the candidate

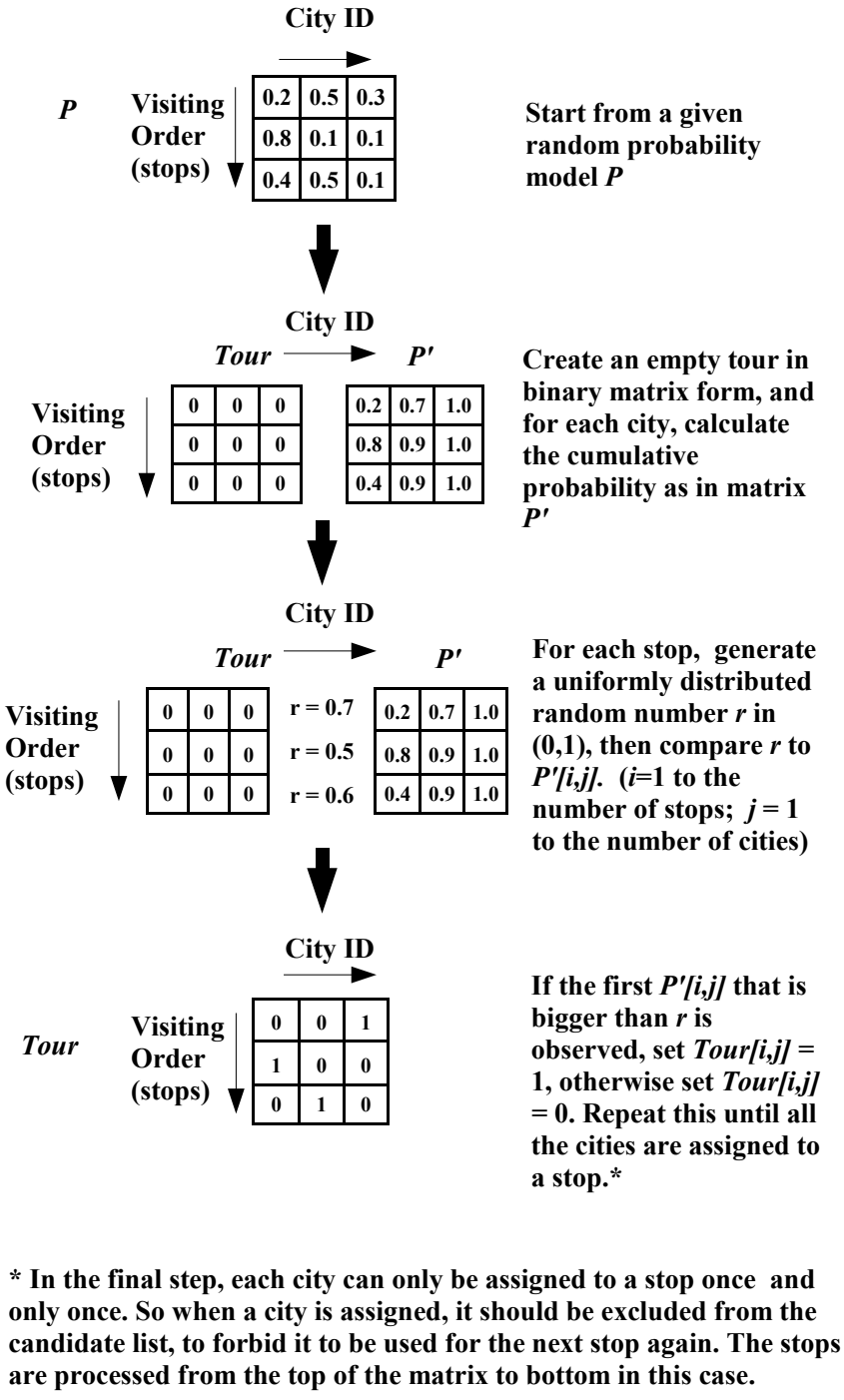


Figure 5.4: An example of sampling new candidate tours from a non-dependent probability model in the EDAs

pool, so that crossover and mutation can pick up and preserve the hidden information in it. On the other hand, in the EDAs, the probability model is used to regenerate new candidate solutions. Hence the current best can be used directly as a perturbation to the probability model, and therefore guide the search with the promising sub-patterns in the current best. This method is presented in the following equation:

$$P_{t+1} = (1 - \alpha) \cdot P_t + \alpha \cdot X + \eta \cdot X_{best} \tag{5.3}$$

where P_t is the probability model; t is the number of the iteration or generation; P_0 is a uniform distribution model; α and η are scalars between $[0, 1]$; X is the extracted information from the selected candidates; and X_{best} is the current best solution. It should be noted that the P, X and X_{best} have an identical data structure. In this case, it is a $N \times N$ matrix, where N is the number of cities.

In the EDAs, this elitism strategy may be regarded as a modified probability model. However, it is better understood as a perturbation to the probability model, which controls the direction of searching in a more quantitative manner. If the current best is far away from the current population, the additional term will drive the centroid of the current population towards the area near the current best; otherwise, it will focus on the current area of the search space.

Combining with Local Search

It is widely recognised that stochastic search algorithms, combined with a proper local search, are a very powerful framework for state-of-practice optimisation algorithm design. Plenty of work in real-world applications has proved successful. Examples include an EDA with the incomplete simplex and the diagonal quadratic approximation local search for continuous problems (Zhang et al. (2004)); an EDA with 2-opt local search for the quadratic assignment problems in Zhang et al. (2006); a GA with a direct local search in Jiang (2003); and a simulated annealing method with gradient descent local search in Pain et al. (1995).

In this project, the analytical problems (TSPs) and the real problem (reactor fuel-management optimisation) belong to the class of combinatorial optimisation problems. The 2-opt or n-opt methods (Buffa et al. (1964)) can be used as the local search method. The 2-opt method is a very well known and understood local search. EDAs combined with 2-opt local search for quadratic assignment problems have been well studied and compared with other algorithms in Zhang et al. (2006). It is widely applicable to most of the combinatorial problems.

Compared to the 2-opt method, gradient-based methods are even better systematically analysed and they have also been combined with the EDAs and applied to a class of continuous problems in which the gradient information is available (Bosman and Thierens (2001)). However, it has not been used to solve combinatorial problems within an EDA's framework. Using the unique formulation of TSPs in Pain et al. (1995), the gradient information can be estimated in TSPs. Therefore, combining the gradient descent method with the EDAs for TSPs is feasible.

In Pain et al. (1995), the objective function (tour length) of a TSP with S cities can be written as:

$$\text{Tour length} = x^T \cdot D_f \cdot x \tag{5.4}$$

where $x^T = (x^1, x^2, ...x^S)$, and x_i^u is the probability that city i is in position u of the tour. Matrix D_f consists of $S \times S$ blocks of matrices $D_f(i,j)$ where

$$D_f(i,j) = \begin{cases} D & \text{if } j = i+1; \text{ or } i = S \text{ and } j = 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

The matrix D is a $S \times S$ distance matrix, $D(i,j)$ being the distance from city i to city j.

For example, a four-city TSP problem that has three cities on a straight line can be presented as:

$$\{c_1 \leftrightarrow c_2 \leftrightarrow c_3 \leftrightarrow c_4\} \tag{5.6}$$

The distance between any two adjacent cities is one unit. The corresponding distance matrix D is:

$$D = \begin{bmatrix} 0,1,2,3 \\ 1,0,1,2 \\ 2,1,0,1 \\ 3,2,1,0 \end{bmatrix} \tag{5.7}$$

The corresponding D_f is:

$$D_f = \begin{bmatrix} 0,D,0,0 \\ 0,0,D,0 \\ 0,0,0,D \\ D,0,0,0 \end{bmatrix} \tag{5.8}$$

Another form of D_f, D_b, is defined as follows:

$$D_b(i,j) = \begin{cases} D & \text{if } j = i-1; \text{ or } i = 1 \text{ and } j = S \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

Using the above example, D_b is:

$$D_b = \begin{bmatrix} 0,0,0,D \\ D,0,0,0 \\ 0,D,0,0 \\ 0,0,D,0 \end{bmatrix} \tag{5.10}$$

Then another form of tour length can be written as:

$$\text{Tour length} = \frac{1}{2} \cdot x^T (D_f + D_b) \cdot x = \frac{1}{2} x^T D_s x \tag{5.11}$$

where D_s is symmetrical.

Alternatively, a dual problem can be used to maximise the sum of the length

of all the edges of the complete graph except the chosen tour.

$$F = \frac{1}{2} x^T G x \tag{5.12}$$

where

$$G(i,j) = \begin{cases} D & \text{if } j = i+1 \text{ or } j = i-1; \\ 0 & \text{if } j = i \\ 2D & \text{otherwise} \end{cases} \tag{5.13}$$

If $i+1 > S$ then set $i+1 = 1$, if $i-1 < 1$ then set $i-1 = S$. In the above four-city example, G is:

$$G = \begin{bmatrix} 0, D, 2D, D \\ D, 0, D, 2D \\ 2D, D, 0, D \\ D, 2D, D, 0 \end{bmatrix} \tag{5.14}$$

The first derivative of the objective function is Gx. Note that there are strong constraints that must be satisfied:

$$\sum x_i^u = 1 \;\; (i \;\; = 1 \text{ to } S \text{ and } \;\; u = 1 \text{ to } S) \tag{5.15}$$

A standard gradient descent local search (Snyman (2005)) described in equation 5.16 can be implemented using the gradient information of the current best tour. The local search continues until the local optimum is reached or, alternatively, a better tour is found.

$$X'_{best} = X_{best} + \gamma \cdot g \tag{5.16}$$

where g is the gradient of the object function at X_{best}, which is $G \cdot X_{best}$, and γ is a scalar.

To simplify the complexity of the algorithm, we use an alternative strategy for using the gradient in the TSP. Instead of implementing another search method,

we combine the gradient information into the probability model itself, so that the sampling operator is able to generate new candidate tours in that direction. It is reasonable to consider this local search as another perturbation to the probability model.

Hence a modified probability updating equation can be established:

$$P^{(t+1)} = (1 - \alpha) \cdot P^{(t)} + \alpha \cdot X + \eta \cdot X_{best} + \gamma \cdot g \tag{5.17}$$

where the P is the probability model used in the EDA, α, η and γ are scalars, X_{best} is the current best and g is the gradient of the X_{best}.

By doing this, the probability sampling operator will be able to produce candidates toward the steepest descent direction of the current best tour, but without losing the information learnt from other candidate tours. Another advantage is that the current best and its gradient are separated and can be customised when necessary.

Combining with Heuristic Information

Using problem-dependent heuristic information can help the optimisation algorithm to improve its performance, especially the black-box type of methods. Examples include the use of a distance matrix in the Ant Colony Optimisation (ACO, Dorigo and Gambardella (1997)) and a multi-objective EDA with problem-dependent regularity information (Zhang et al. (2008)).

The distance between cities in the TSPs is useful and makes a direct contribution to the objective function - the tour length. We use this information in the proposed EDAs to improve its performance. The method is from the ACO (Dorigo and Gambardella (1997)). Using the data structure of the probability model and the tour sampling operator in figures 5.4 and 5.3, it is straightforward to combine the distance information into the probability model in order to guide the tour regeneration. The probability model is perturbed in the sampling operator as follows:

$$P_{sampling}(i,j) = \frac{P(i,j)}{D(k,i)^{\beta}} \tag{5.18}$$

where $P_{sampling}$ is the probability used to sample new tours, P is the probability updated by the standard updating operator shown in figure 5.3, D is the distance matrix, $D(m,n)$ is the distance from city m to city n, i is the visiting order or the stops, j is the city IDs, k is the ID of the city visited at the last stop and β is an exponential scalar.

The above method is incorporated in the sampling operator. The idea is that the choice of the next city to visit depends on how often it is used in known promising tours (see figure 5.3), as well as on the distance between the candidate cities and the last visited city. By using this sampling probability, the length of sub-paths is combined into the EDA.

For example, a new tour is being generated for a 3-city TSP. The distance between city 1 and 2, and between 2 and 3, is one unit, and the distance between city 1 and city 3 is two units. The way in which the distance information is used in the EDAs is illustrated in figure 5.5.

5.2.4 Summary

In this section we discussed the performance-enhancement methods that are available in the EDAs for the analytical TSPs, including the use of the elitism strategy, the use of simplified gradient-based local search and the use of the problem-dependent heuristic information - the distance matrix in TSPs.

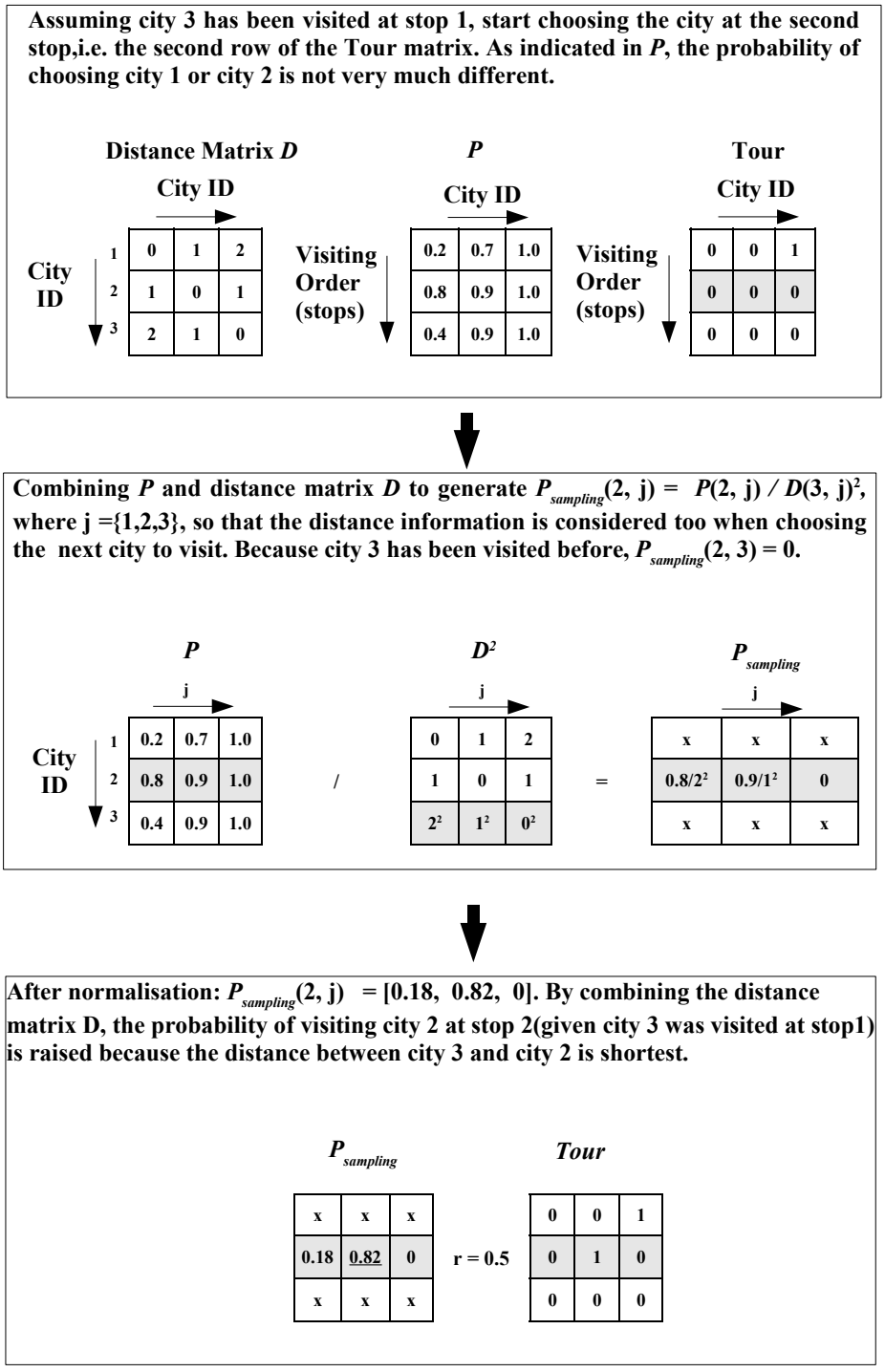


Figure 5.5: An example of combining the heuristic information with the probability model in the EDAs for TSPs: generating a candidate tour with population-based learning and the distance matrix.

5.3 Performance Comparison of the EDAs and Benchmark Algorithms

5.3.1 A Set of Analytical TSPs

A set of eight Travelling Salesman Problems are used as analytical problems to demonstrate EDAs' performance compared to classic GAs with well-established crossover operators.

Problems 1, 2 and 3 are taken from the paper by Poon and Carter (1995), Problems 4, 5, and 8 are taken from the TSPLIB website maintained by Gerhard Reinelt (Reinelt (1997)). Problems 6 and 7 are from Whitley et al. (1989). The problems are briefly described below:

Problem 1: A 20-city Hamiltonian Path TSP.
In this case, the salesman can start from any city, and each city should be visited exactly once. All the cities lie on a straight line. City c_i is accessible from its two directly adjacent neighbours only. The distance between any two adjacent cities is one unit. Problem 1 is illustrated below:

$$\{c_1 \leftrightarrow c_2 \leftrightarrow c_3 \leftrightarrow ... \leftrightarrow c_{20}\} \tag{5.19}$$

There are two equivalent optimal tours with a length of 19 units; they are:

$$\begin{aligned} &\text{Optimal Tour A: } \{c_1 \rightarrow c_2 \rightarrow c_3 \rightarrow ..., \rightarrow c_{19} \rightarrow c_{20}\} \\ &\text{Optimal Tour B: } \{c_1 \leftarrow c_2 \leftarrow c_3 \leftarrow ..., \leftarrow c_{19} \leftarrow c_{20}\} \end{aligned} \tag{5.20}$$

Problem 2: A 20-city Hamiltonian Cycle TSP.
The salesman can start from any city and go back to the first one when all the cities are visited. All the cities lie on a circle. It is very similar to Problem 1, except that there is a direct two-way path between the first c_1 and the last city c_{20}, with a

length of one unit.

$$\{c_1 \leftrightarrow c_2 \leftrightarrow c_3 \leftrightarrow ... \leftrightarrow c_{19} \leftrightarrow c_{20}\} \text{ and } \{c_1 \leftrightarrow c_{20}\} \tag{5.21}$$

There are forty equal optima with a tour length of 20 units. An optimal tour may start from any city and travel along either direction of the circle until the first visited city is reached again, e.g.

$$\{c_5 \rightarrow c_6 \rightarrow c_7 \rightarrow ... \rightarrow c_{19} \rightarrow c_{20} \rightarrow c_1 \rightarrow c_2 \rightarrow ... \rightarrow c_5\} \tag{5.22}$$

Problem 3: The classical Oliver 30-city Hamiltonian Cycle TSP.
A well-known TSP that has been used in testing GAs in Oliver et al. (1987). There are sixty equivalent optima, with a tour length of 424 units.

Problem 4: A symmetrical 16-city TSP named Ulysses16 from TSPLIB. The length of the best known tour is 68.59.

Problem 5: A symmetrical 52-city TSP named Berlin52 from TSPLIB. Information of 52 places in Berlin are transferred to this TSP. The length of the best known tour is 7542.

Problem 6: A symmetrical 50-city TSP named Eil50 from Whitley et al. (1989). The length of the best known tour is 428.

Problem 7: A symmetrical 75-city TSP named Eil75 from Whitley et al. (1989). The length of the best known tour is 545.

Problem 8: A symmetrical 101-city TSP named Eil101 from TSPLIB. The length of the best known tour is 629.

5.3.2 Algorithms Implementation

We implement the following EDAs for the analytical TSPs:

EDA_elitism

An EDA very similar to the standard Population-Based Incremental Learning (PBIL) algorithm, but with two modifications. The conventional elitism strategy is used, which is to preserve the current best in the population all the time. In addition, the new elitism strategy proposed in equation 5.3 is also used.

EDA_gradient

An EDA built on top of the EDA_elitism. The elitism strategy that perturbs the probability model directly is used. In addition, the gradient information of the current best tour is combined with the probability model. Details are described in the previous section.

EDA_heuristic

An EDA built on top of the EDA_elitism. The distance information between cities is used when generating new candidate tours.

In all the EDAs, a swap mutation (Oliver et al. (1987)) is also used with a low rate. Although it is not a standard component of the EDAs, a conventional mutation operator can be used if desired, or even advanced mutation operators like the guided mutation (Zhang et al. (2005)). Similar to GAs, the swap mutation operator in the EDAs is applied to individual candidate solutions, not the probability model.

The results of the proposed EDAs are compared against three genetic algorithms researched in Poon and Carter (1995). We choose these three because they have given good results more consistently compared to others. The crossover operators used in the GAs are:

GA_CX

A GA with the Cycle Crossover (CX) by Oliver et al. (1987);

GA_OX

A GA with the Order Crossover (OX) by Oliver et al. (1987);

GA_UX2

A GA with the Union Crossover #2 (UX2) by Poon and Carter (1995).

SA

A standard simulated annealing algorithm, which starts with a randomly-chosen tour. Two cities that are also randomly chosen swap their positions to produce a new solution. If the generated solution has a shorter length then it is accepted, otherwise, it is accepted based on a probability of acceptance:

$$P_{t+1} = e^{\frac{f_t - f_{t+1}}{T}} \tag{5.23}$$

where P_{t+1} is the probability of accepting the current tour at step $t+1$; f_t is the tour length at step t; T is a scalar parameter called the temperature. The initial temperature T_0 is normally set to a large value. T is decreased if and only if a better solution is found, using the following equation:

$$T_{t+1} = \omega \cdot T_t \tag{5.24}$$

From experimental results in the next sub-section, this implementation of SA works well when applied to small and medium sized TSPs, as shown in our experiments.

To validate the comparison, all algorithms are tested twenty-five times independently with the same parameters but different random seeds. The algorithms' parameters settings are summarised in table 5.1. All the algorithms use small populations and generations compared to what can be used in the real-world applications.

Algorithms	Population Size	Max Generations	Mutation Rate	Crossover Rate
EDA_elitism	21	300	0.05	N/A
EDA_gradient	21	300	0.05	N/A
EDA_heuristic	21	300	0.05	N/A
GAs	21	300	0.2	0.8

Algorithms	α	η	γ	β
EDA_elitism	0.001	0.01	N/A	N/A
EDA_gradient	0.001	0.01	0.001	N/A
EDA_heuristic	0.001	0.01	N/A	5

Algorithms	T	ω	Maximum number of function evaluations
SA	2000	0.97	6300

Table 5.1: Parameter settings for the tested EDAs, GAs and SA for the analytical TSPs

The algorithm parameter settings are not intended to be optimal. The reason for such an experimental setting is to demonstrate the robustness of the tested algorithms under an unknown situation, and to validate their performance given a limited number of objective function evaluations, which is often a realistic constraint in real-world applications in which the objective function evaluation is extremely expensive in terms of the computational time, an example being the reactor loading pattern simulation.

We tuned the α, η and β and the mutation rate used in the EDAs where applicable, in order to achieve reasonable quality and to study the potential of the algorithms given a limited number of objective function evaluations.

5.3.3 Experiments Results

We set up two experiments to study the algorithms. In Experiment 1, we aim to understand the performance of the basic EDAs with simple and well-understood test problems. For this purpose, the first three analytical TSPs are chosen. Only the EDA_elitism and EDA_gradient are tested because they did not use the heuristic information (or not directly). The results are shown in table 5.2.

Algorithms	Problem 1		Problem 2		Problem 3	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
EDA_elitism	37.88	7.37	27.42	6.87	562.43	34.94
EDA_gradient	35.80	7.30	30.57	5.64	559.78	37.01
GA_CX	48.28	8.31	36.62	3.62	692.67	58.29
GA_OX	45.88	9.37	35.60	4.82	676.24	61.50
GA_UX2	42.64	3.72	36.36	4.19	679.59	45.99
SA	25.88	5.30	23.08	5.66	502.19	29.52
Best Known	19		20		424	

Table 5.2: Experiment 1 results of applying the EDAs to the analytical TSPs comparing to the GAs and SA. Mean is the averaged tour length of the twenty-five best found in twenty-five independent runs and S.D. is their standard deviation.

In Experiment 2, the three implemented EDAs are tested on other TSPs, which are derived from real-world locations or from published literature. This experiment aims to gain an understanding of the potential of the EDAs when applied to problems that are not well understood, and of how the EDAs can be improved by using the practical performance-enhancement methods. The results are shown in tables 5.3.

Discussion

From the results of Experiment 1, we see that the two EDAs perform slightly better than the three GAs. Among the tested GAs, the performances are not significantly different. The SA outperforms all other algorithms. In Poon and Carter (1995), a set of GAs with the same crossover operators as used here were applied to Problem 1 and 2, and their results are better. This is due to the implementation details of the computer programs.

In Problem 3, the EDA_gradient has slightly better averaged results but a larger standard deviation than the EDA_elitism. This is because the gradient information helps local convergence, but may also more easily trap the search at a poor local optimum. If the algorithm was able to locate an area containing a near-optimal solution, then the gradient information would help the convergence. Otherwise, the gradient may lead to a poor solution that is also a local optimum. So the variance of a number of independent runs is likely to be larger, particularly when the search

Algorithms	Problem 4		Problem 5		Problem 6		Problem 7		Problem 8	
	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
EDA_elitism	75.42	2.10	11838.41	653.41	657.82	30.21	1060.26	67.10	1488.11	74.47
EDA_gradient	76.61	3.53	18334.26	1155.97	N/A	N/A	N/A	N/A	N/A	N/A
EDA_heuristic	75.04	0.75	9108.50	507.98	487.56	12.28	655.66	22.02	823.76	48.68
GA_CX	78.42	3.61	16743.60	880.47	924.79	50.15	1532.71	53.41	2199.29	97.36
GA_OX	82.13	3.62	16513.46	710.17	925.11	58.17	1518.95	79.93	2183.11	67.44
GA_UX2	77.78	2.60	16896.13	866.73	928.47	49.92	1548.27	74.59	2240.46	76.02
SA	74.55	0.58	10212.00	571.93	587.33	35.40	857.73	40.90	1179.80	65.04
Best Known	68.59		7542		426		538		629	

Table 5.3: Experiment 2 results of applying the EDAs to the analytical TSPs comparing to the GAs and SA. Mean is the averaged tour length of the twenty-five best found in twenty-five independent runs and S.D. is their standard deviation.

space is large.

The standard deviations of the EDAs are considerably smaller compared to the test GAs in Problem 3. As pointed out in Poon (1992) and Poon and Carter (1995), crossover operators in the GAs can be very disruptive, e.g. the UX2, because it tends to retain and pass on some sub-patterns, which are normally short, from the promising tours to the next iterations. It is also noted that the UX2 may be likely to break the longer promising sub-patterns that are near-optimal (Poon (1992)). In the EDAs, the probability model and the sampling are able to keep very short schemata (such as the position of a single city) naturally, without being very disruptive.

It is not surprising that the tested SA produces better results than other algorithms in Experiment 1. Unlike the very disruptive GAs, SA turns into a greedy search gradually during the search process. In terms of local convergence, the SA is the best, as expected. The GAs, on the other hand, are exploring a wider range of solutions than the SA and EDAs, but not able to converge very well. The EDAs' results show a better convergence than the GAs, as well as a noticeably larger variance than the SA.

For Problems 1 and 2, the SA is able to converge on near-optimal solutions. There are a large number of local minima in these two problems, but if a good solution beyond the local minimum is found, the SA can easily solve them simply by performing a greedy search. The EDAs and GAs do not have such a mechanism, and therefore they did not perform particularly well on these two problems. Because the sampling operators in the EDAs are trying to generate more solutions that are similar to the current best, they are less disruptive than the GAs. Hence they achieved better averaged results and smaller standard deviations from a number of independent runs. The same pattern can be observed in Problem 3, although due to a much larger search space, none of the algorithms perform particularly well given a limited number of objective function evaluations and non-optimal parameters.

In Experiment 2, the EDA_gradient is only applied to Problems 4 and 5 because of its poor performance. The local search feature in the EDA_gradient may

prevent necessary explorations and its standard deviation is also considerably larger, as explained in the discussion of Experiment 1. The EDA_heuristic outperforms the other two EDAs significantly because the distances contribute directly to the objective function. The probability model provides a flexible vehicle with which to combine such information.

The tested GAs in Experiment 2 are more varied. Their performance is not as promising, because it is expected that the crossover operators in these algorithms are very disruptive and can be sensitive to the parameter settings, such as the number of function evaluations and population size, and implementation details.

In Problem 4, the SA outperforms tested GAs significantly, and is also slightly better than the EDAs, because it converges better when the problem size is relatively small. In Problem 5, the SA is still better than the tested GAs but is outperformed by EDA_heuristic. It is clear that when the problem size is large, a local search method can easily be trapped in a local optimum and not be able to explore the search space, the EDA_gradient being another example. The GAs and the EDA_elitism does not perform as well as EDA_heuristic in this case, but they all have larger standard deviation values, suggesting that they are exploring more widely in the search space.

Results for Problems 6, 7 and 8 show a similar pattern, namely, that the tested SA is a better local searcher, with some difficulties in exploring a large search space; the GAs, on the other hand, explore well but do not exploit efficiently. The EDA_elitism shows a 'balanced' exploration performance in terms of convergence when compared to the SA and GAs. The EDA_heuristic outperformed others considerably. However, a very fast convergence sometimes suggests that the results might not be improved significantly afterwards.

From table 5.3, none of the algorithms perform particularly well considering the quality of the best known solutions. It should be noted that the results are the averaged best solutions from 25 independent runs. Among these runs, some high quality solutions were found, but it is hard to judge a stochastic method by showing

Problem ID	Pop. Size	Max. Gen.	Mutation Rate	α	η	β
Problem 6	21	300	0.02	0.001	0.01	10
Problem 7	21	300	0.01	0.001	0.01	7

Table 5.4: Slightly tuned parameters for EDA_heuristic applied to Problem 6 and 7.

Algorithms	Problem 6			
	Best	Gap	Mean	No.Obj.
SA	443	3.9%	N/A	68512
EDA_heuristic	440.42	3.4%	464.61	6300
Best Known	426			

Algorithms	Problem 7			
	Best	Gap	Mean	No.Obj.
SA	580	7.8%	N/A	173250
EDA_heuristic	572.07	6.3%	597.02	6300
Best Known	538			

Table 5.5: The best solutions found by a well-designed and tuned SA applied to Problem 6 and 7 in Lin et al. (1993) compared to the EDA_heuristic. EDA_heuristic is tested with 25 independent runs and slightly tuned to give better results than in table 5.3. The percentages in brackets show the gap between the best found and the best known solutions - Gap = $\frac{\text{Best found - Best known}}{\text{Best known}}$.

the best solution that it once reached. Therefore, averaged results with standard deviation are used for demonstrating the algorithms' robustness and reliability. Also the algorithms in this chapter are not designed to be the state-of-the-art TSP solvers but to understand them and to demonstrate how they work. For this reason, the identical parameters settings for all the EDA algorithms and test problems are not optimal and the maximum number of objective function evaluations is very limited.

Having said that, the EDA_heuristic has shown that it is able to converge to solutions with reasonable quality, given a very limited number of objective function evaluations. It might be worthwhile to see its performance with reasonable parameter-tuning and compare it with a good benchmark.

In Lin et al. (1993), a well designed SA is applied to Problem 6 and 7. We applied the EDA_heuristic algorithm again to these two problems with slightly tuned parameter settings, shown in table 5.4. The results summarized in table 5.5 show

that the EDA_heuristic produces very similar, if not better, solutions compared to the benchmark SA, the best found solutions are within a 3.4% and 6.3% gap of the best known ones. Meanwhile, the number of function evaluations used by EDA_heuristic to achieve this is only 6300 for both cases - approximately 10% and 4% of the SA's.

Considering all the results in Experiment 2, the EDA_heuristic consistently finds reasonably good solutions with a considerably small number of objective function evaluations. This is one of the most desirable features of the evolutionary algorithms. However, the main concern is that it may not be able to explore the whole search space thoroughly because of its fast convergence, due to the heuristic information used.

It should be noted that the sampling operator used in the proposed EDAs has a time complexity of $O(n^2)$ (n is the number of cities), which is more expensive than a normal GA crossover operator. In real-world applications, especially those having very time-consuming objective function calculations, the cost of using complicated operators is negligible and these can be used freely (Poon (1992)).

However, this only applies to the solution-reproducing operator. In the EDAs, if an advanced probability model is chosen that needs to be re-built or updated in every iteration, the computational cost can be too high to accept - it grows exponentially with problem size, as discussed in section 4.5.2. In fact, looking for the complete/optimal variable dependencies given a set of samples is a hard optimisation problem in its own right. Therefore, the complexity of the EDAs' sampling operator is acceptable with most of the applications, but the type of probability model should be considered carefully. In this case, we used a model without variable dependency.

The EDAs are generally recommended for similar problems because of their better performance, subject to further improvement. In some GAs' applications, a specially-designed crossover operator is required, while the EDAs do not have such problems, provided that the problem is properly encoded.

5.4 Summary

In this chapter, we used the TSPs as the analytical problems in order to understand and demonstrate the potential of the proposed EDAs in solving this classical combinatorial optimisation problem. The results are compared to three GAs with different crossover operators and a SA algorithm used in previously published research.

The permutation integer vector and the alternative binary matrix encoding method are introduced to represent the candidate solutions in the EDAs for TSPs. Like the conventional GAs, the natural encoding is preferred in the EDAs in order to achieve better performance.

Unlike the GAs, the solution-regenerating component in the EDAs is based on a probability model suggesting the distribution of the promising solutions, which is dynamically updated during the search. The new candidate solutions are sampled from the model. Despite the variety of the available types of the probability models, we suggest using a relatively simple, but robust probability model because of its computational efficiency and flexibility for further improvement, which is also recommended by previously published research studies on EDAs.

Using a probability model to guide the search confers flexibility to the algorithm, which can be improved with little effort by adding any relevant information. EDA_heuristic is an example of this.

The experimental results of the EDAs on a set of TSPs show that a standard EDA's performance is as good as or better than the GAs with some of the well-recognised crossover operators on TSPs. This is mainly because the EDAs replaced the crossover operator with the probability model sampling routine, so the search is expected to be less disruptive than the GAs. The EDA_heuristic outperformed the tested GAs and the SA in some cases. A more important point is that this EDA seems to be able to find reasonable solutions with limited cost, which is the main drive for the applications of evolutionary algorithms. When the objective function evaluation is very difficult or computationally expensive, this feature is often desirable.

Chapter 6

Estimation of Distribution Algorithms for Reactor Fuel Management Optimisation

In this chapter we discuss how EDAs can be applied to reactor fuel-management optimisation problems. The key considerations are the LP representation (encoding) and how to adapt EDAs to this class of problems.

We introduce a binary matrix LP representation method first, and then describe the EDA-based optimisation algorithm used in this work. Finally, we suggest a generic framework through which heuristic information can be combined in EDA-based optimisation algorithms as a practical option for improving their performance.

6.1 Loading Pattern Representation

An LP of a reactor can be regarded as a set of assignments of n items to m positions, where n is the number of fuel types or fuel elements and the m positions could include in-core fuel channels and out-of-core store locations. In this work, we consider that the basic building block of an LP is a fuel ID - fuel channel assignment. Since we already have a list of fuel IDs and a list of in-core fuel channels, this suggests that we use a binary matrix to represent the whole in-core LP.

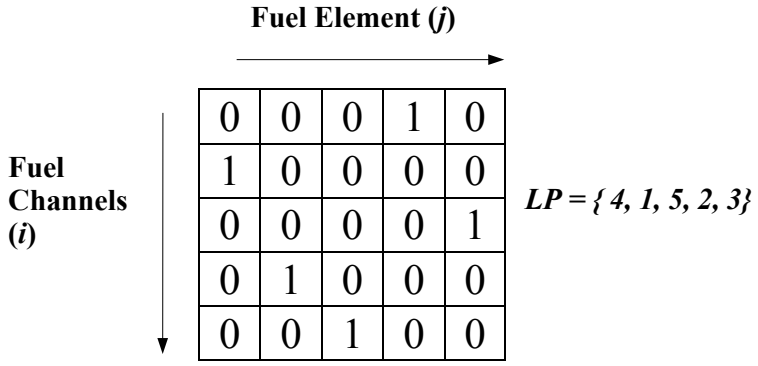


Figure 6.1: A binary matrix representation of a reactor loading pattern

In a binary matrix, with N rows and M columns, each row of the matrix represents an in-core fuel channel, while each column represents a fuel element ID or fuel type ID. Entry $LP(i, j)$ is set to 1 if and only if fuel channel i is loaded with fuel ID j, otherwise $LP(i, j)$ is 0. Since one channel can only be occupied by one fuel element, there is only one 1 on each row.

It can be argued that when the number of fuel elements is very large, this sparse matrix representation may not be computationally efficient and obviously costs more storage. However, we can always transform such a binary matrix into an integer vector, as shown in figure 6.1, without losing any information. We will use the binary matrix form because it is better for explaining the element-channel assignment concept and the optimisation algorithms.

6.2 The Algorithm

Based on the probability-model-driven mechanism of EDAs, a customized EDA algorithm for reactor LP optimisation is presented here. It collects the statistical information of how often a certain fuel-channel assignment appears in relatively

good LPs, and uses this information to search for better LPs. An initial probability model is maintained, evolved and perturbed during the optimisation process. New LPs are generated by sampling this model. The algorithm is described as follows:

(1) initialize a population of LPs randomly or by sampling an initial probability distribution. Refer to figure 5.4 for the sampling method;

(2) select some individuals using a selection method based on a pre-defined objective function;

(3) update the probability model using the selected LPs and the following equations from Baluja (1994), the Population-Based Incremental Learning (PBIL) algorithm.

$$P_{t+1} = (1 - \alpha) \cdot P_t + \alpha \cdot X \tag{6.1}$$

where P is the probability model for core channels, α is a scalar between [0,1] and X is the statistical information extracted from the selected LPs. An example of learning and updating the probability model is illustrated in figure 5.3;

(4) Sample new LPs from the updated probability model; a mutation operator can be used if desired. An example of the sampling method is illustrated in the following figure. Examples of updating and sampling are given in figure 5.3 and figure 5.4, respectively;

(5) If the stop criterion is not met, then go back to step (2).

6.3 Using Heuristic Information in EDAs

In real-world applications, using heuristic information in black-box optimisation has been regarded as a successful enhancement method (Zhang (2003) and Zhang et al. (2004)). An example in reactor fuel-management optimisation can be found in Poon and Parks (1993). It does not guarantee that the global optimum will be reached. Heuristic information is used to find a near-optimal or acceptable solution in reasonable time. This is the main drive for the application of heuristic information in this work. The key questions are how to extract it from a given problem and how to use it in optimisation algorithms.

In the algorithm described, the LP optimisation is a process of choosing and recombining the basic building blocks - the assignment of fuel elements to fuel channels. In a normal EDA-based algorithm, it is done by sampling a probability model that is being updated by statistical information collected from evaluated candidate LPs.

Let us assume that there is some information, H, for each building block indicating its contribution to a pre-defined objective function. The greater the contribution made by a building block, the more likely it is that it will appear in the optimal solution. How we generate the H for reactor fuel-management optimisation will be discussed in the next chapter.

To use this H in EDAs, we suggest using the same matrix structure as in the binary matrix LP representation and the probability model P. By doing so, the heuristic information H can be easily used to perturb the probability model P. The modified algorithm is similar to the one described earlier in a previous section. The only difference is that, when sampling new LPs, a perturbed P^h is used instead of the original P.

$$P^h = P \cdot H^\beta \tag{6.2}$$

where H contains heuristic information, β is an exponential scalar adjusting the

weight between population-based learning and heuristic information. An example of using H in EDAs is illustrated in figure 6.2.

Using this idea, any useful heuristic information, expert knowledge or even random perturbations can be used to direct the optimisation to concentrate the search on some particular area without sacrificing too much exploration.

6.4 Summary

In this chapter, we described how to set up EDAs to solve reactor loading-pattern optimisation problems. A binary matrix encoding method is introduced to represent the loading patterns, of which the basic building block is the 'fuel element - fuel channel' assignment. An EDA-based algorithm is then presented, followed by a further extension of how to combine heuristic information with it.

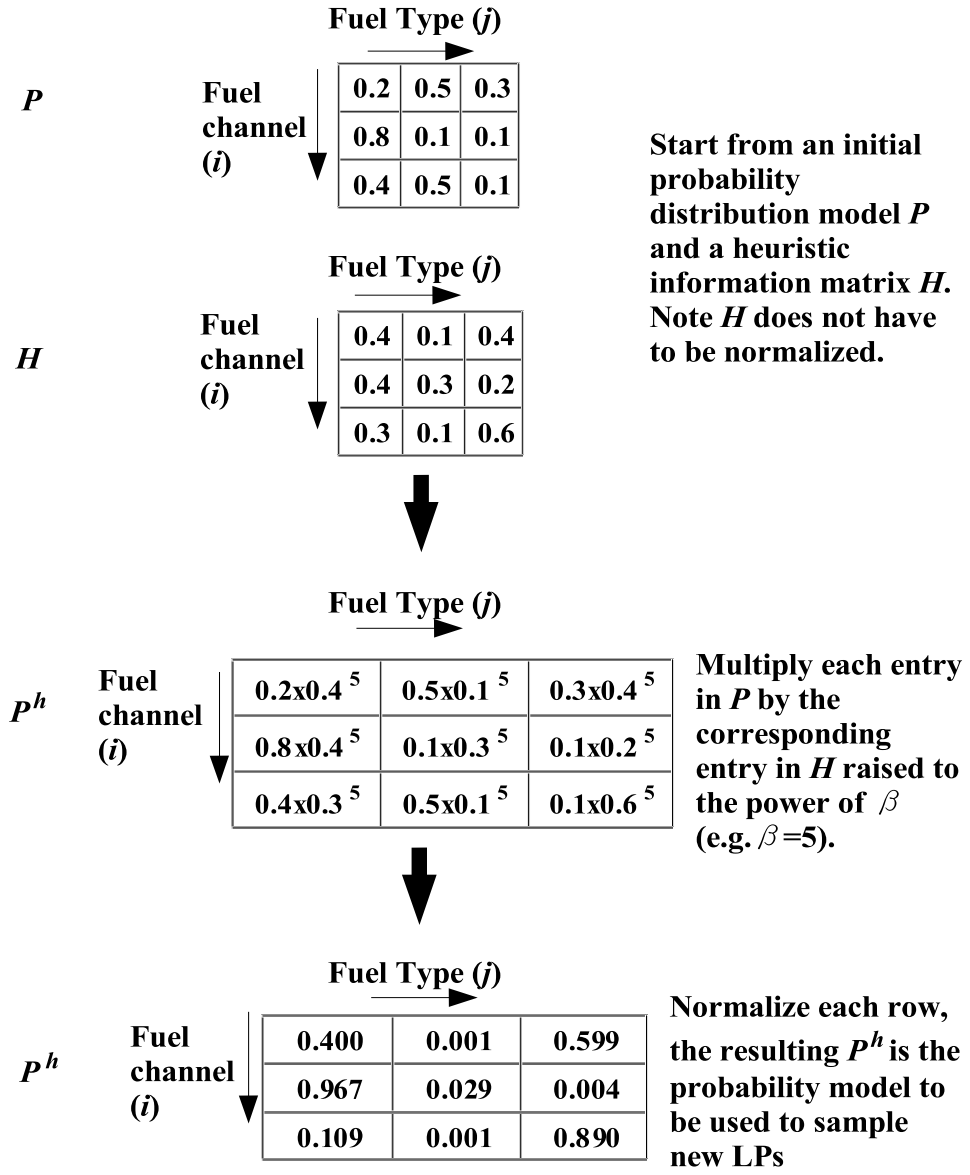


Figure 6.2: Combining heuristic information with the probability model used in EDAs

Chapter 7

Applications and Results

In the previous chapter, we explained how to use EDAs to solve an example reactor loading pattern optimisation problem. In this chapter, we set up realistic reactor fuel management optimisation problems based on the Imperial College CONSORT Reactor (the CONSORT reactor), and apply our proposed EDAs to them. The resulting algorithms are analysed and compared to a number of well-studied GAs with various crossover operators, which have been proved to be robust and efficient in many engineering applications, including reactor fuel management optimisation.

We chose the CONSORT reactor as the test bed, because while it is a small research reactor, the principle of loading pattern optimisation is the same as for much larger commercial reactors. The optimisation method applied to the CONSORT reactor can be applied to others without difficulty.

We start with a description of the CONSORT reactor and derive three test cases from it. These three test cases represent basic examples of typical situations in reactor fuel loading optimisation problems.

When the number of fuel elements or assemblies is too large to be explicitly encoded in an optimisation algorithm - as it might be in a relatively large nuclear power plant - the fuel elements will need to be categorised into a few different types, which means a smaller number of fuel types are used instead of a huge number of fuel elements with similar properties that do not contribute much to loading pattern optimisation.

Hence the first test case is designed to cover this situation. We use a hypothetical and relatively large fuel store in the CONSORT reactor. The fuel elements are then divided into a few fuel types, which reduces the problem size considerably.

The second test case covers the situation in which the number of fuel elements is small enough, or there is an extremely high accuracy requirement, so that each fuel element may need to be encoded into the optimisation algorithm explicitly. The third test case is similar to the second one, but with a power peaking constraint to create a more complicated case, and to test the EDAs with a constrained problem.

We apply the developed EDAs and benchmark GAs to the test cases and compare their performances. To be able to carry out this work, we set up the following sub-projects:

(1) development of simulation models for the CONSORT Reactor using the in-house code EVENT and GEM (de Oliveira et al. (2001));
(2) construction of a set of problem-dependent heuristic data;
(3) development of a fast reactor parameter prediction tool using neural networks;
(4) implementation of a number of GAs with various crossover operators as benchmark algorithms for comparison study, particularly the HTBX (Poon (1992)).

These sub-projects are very important to this research. However, in this thesis, we will concentrate on the optimisation algorithms and only explain these sub-projects briefly. Also, a parameter sensitivity study for the EDAs applied to LP optimisation is presented, followed by some preliminary experiments for applying the EDAs to multi-objective problems.

Fuel Name	MARK I_A	MARK I_B	MARK I_C	MARK II	MARK III
K_∞	1.66197	1.16471	0.79238	1.77473	1.78583

Table 7.1: K_∞ of five fuel types of a hypothesis test case from the Imperial College CONSORT Reactor, calculated using the WIMS code

7.1 A Brief Description of the Imperial College CONSORT Reactor

The Imperial College CONSORT Reactor was first constructed in 1965, and was subsequently expanded in 1971; it has been in continuous safe operation since 1965 at the Imperial College Reactor Centre (the reactor centre). It is the only civil research reactor in the UK. The CONSORT reactor is designated as a low-power research reactor to provide the neutral particles, or neutrons, that result from nuclear reactions inside the reactor core for research and engineering applications.

The reactor centre provides facilities for the university and other educational institutions to be used for teaching and research in many fields of nuclear science and technology, such as reactor physics, reactor engineering, neutron physics, solid-state physics, radiochemistry and activation analysis. The reactor centre also provides radioisotopes for use in other laboratories.

The CONSORT core consists of 24 fuel elements of the MERLIN type that contain highly enriched Uranium fuel plates in an Uranium/Aluminium alloy (Franklin et al. (1998), Franklin et al. (2005)). There are four control rods, rods no. 1, 2, 3 and 4, in the core. Note that rods no. 3 and no. 4 are not included in our modelling, because they act as shut-down rods to ensure safety. Normally they are fully withdrawn. A picture of the CONSORT reactor is shown in figure 7.1 and a 2D core plan is illustrated in figure 7.2.

The fuel plates (or meat) are clad in almost pure aluminium. There are five types of fuel assemblies present in the core of MERLIN design, referred to as MARK I_A, MARK I_B, MARK I_C, MARK II and MARK III. MARK I_A fuel elements are the oldest and contain 12 fuel plates that are slightly curved and were manufactured

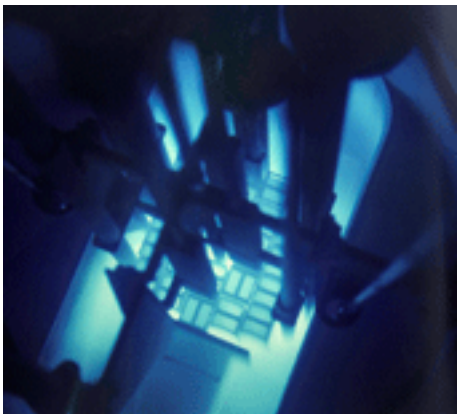


Figure 7.1: A bird's eye view of the Imperial College CONSORT Reactor

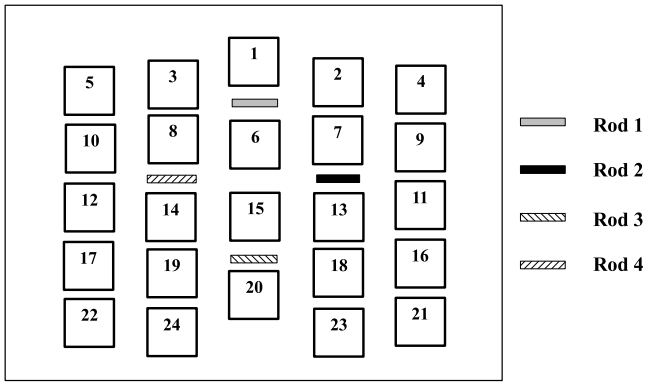


Figure 7.2: A 2D view of the CONSORT reactor (drawing is not to scale)

in 1965. The Mark I_B fuel elements contain 6 fuel plates and the MARK I_C fuel elements contain 3 fuel plates. MARK II fuel elements also have 12 plates, with the fuel meat being thicker than in the MARK I; the plates are not curved and contain about 4g more Uranium metal, and are therefore more reactive than MARK I_A, MARK I_B and MARK I_C plates. MARK III elements have 16 plates that are thinner than the MARK I and MARK II plates. The K_∞ of the five different types of fuel elements are summarised in table 7.1.

The listed fuel inventory information is based on their initial designed status. The amount of uranium has decayed in the last four decades. We used a modified fuel inventory in our test cases by changing the number of fuel elements and/or re-calculating their K_∞.

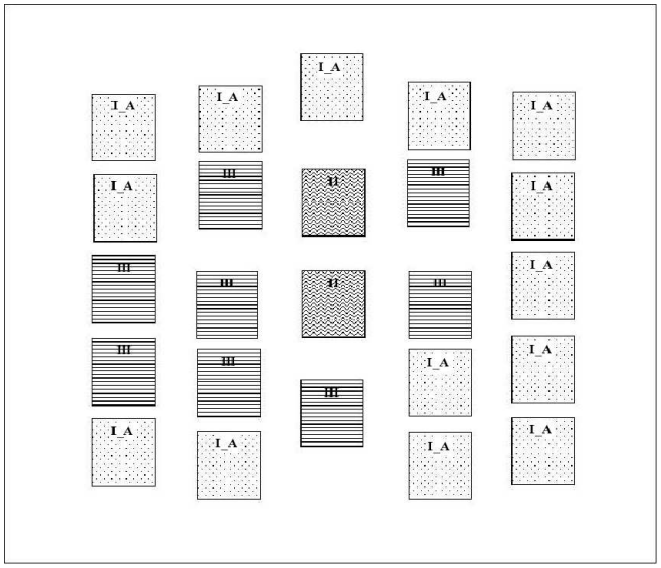


Figure 7.3: An example LP of the CONSORT reactor (drawing is not to scale)

There are only two MARK II fuel elements available and they must be inserted in channel numbers 6 and 15 (see figure 7.2). In order to set up the test case, we will use different inventories for the two test cases. An example of a core loading pattern is illustrated in figure 7.3.

Based on the basic information, a three-dimensional reactor core model was constructed using the radiation transport code EVENT (de Oliveira et al. (2001)). The WIMS8A (SERCO-ANSWERS (1999)) code was used to generate group constants for EVENT. A visualisation of the 3D model is illustrated in figure 7.4.

7.2 Test Case 1 : A Fresh Core

This case represents the situation in which the fuel store is too large, or the difference between fuel elements is too small, so that the fuels have to be grouped into a few types based on some measurement, such as their reactivity. Fuels of the same type are regarded as identical.

Based on the CONSORT reactor, we set up a fresh core test case to start with. 'Fresh' here means we assume all fuel elements are brand new, and all control rods are fully withdrawn. Furthermore, we also removed the aluminium clad on the top

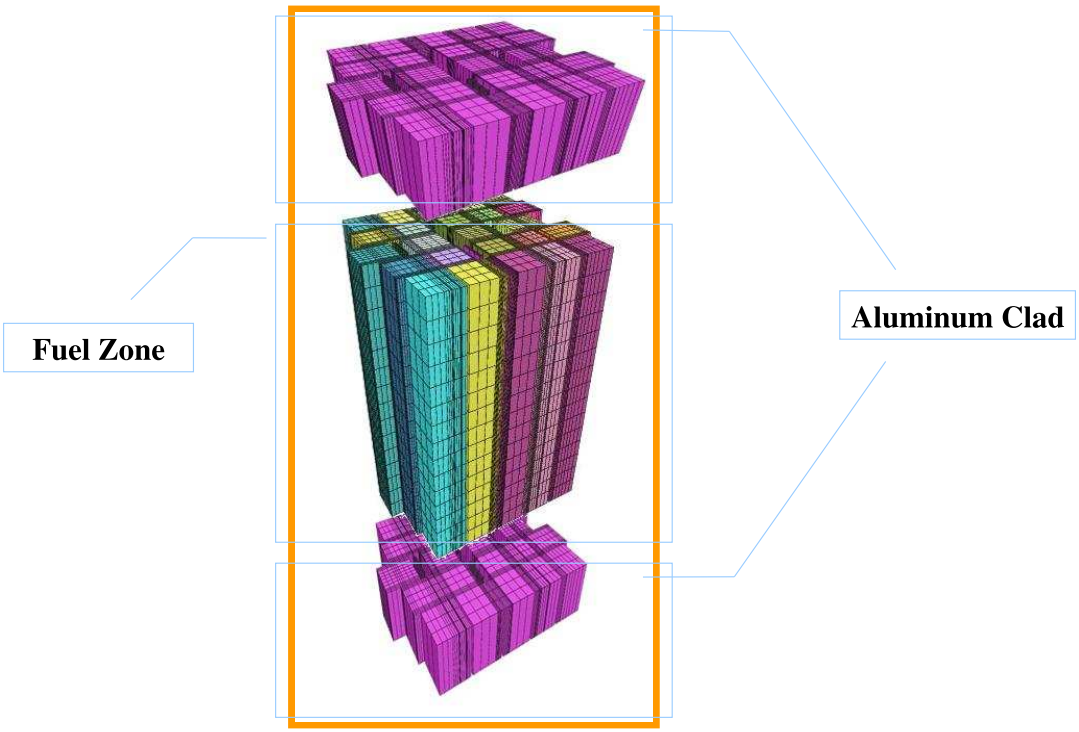


Figure 7.4: A 3D model of the CONSORT reactor by GEM and EVENT, visualised by PLOTTER.

Fuel Type	MARK I_A	MARK I_B	MARK I_C	MARK II	MARK III
Type Code	1	2	3	4	5
No. of Fuel	20	22	22	2	8

Table 7.2: The hypothetical fuel inventory of the CONSORT reactor for Test Case 1. Each fuel type is given an integer code.

and bottom of the fuel elements, as shown in figure 7.4.

By doing so, there are only five different fuel types, and it is also somewhat less complicated because less perturbation or uncertainty is expected, due to the absence of the control rods and metal clad in the core.

7.2.1 The Optimisation Task and the Fuel Inventory

The objective of this test case is to find the optimal LPs for the CONSORT reactor for a given fuel inventory, so as to maximise the effective multiplication factor - the K_{eff}. By doing this, the life of the reactor can be extended, allowing more experiments and research to be performed.

In this case, we make up a fresh fuel inventory with a larger number of fuel elements than the real inventory. Fuel elements are categorised into five different types according to their design, as described in the previous section. We use the initially calculated K_{∞} for all the fuel types in order to form a 'fresh' core state with all the control rods fully withdrawn. The fuel inventory is summarised in table 7.2.

Given the 24-channel CONSORT core and the fuel inventory shown in table 7.2, the search space (number of possible LPs) for this case is approximately 10^{13}. Although 24 fuel channels looks like a small problem, it is still valid for testing the EDAs' performance, because in a typical commercial Pressurised Water Reactor (PWR), one-fourth or one-eighth core symmetry can be applied, which greatly reduces the search space to a similar size. For example, the number of fuel channels considered in Biblis PWR reloading problem from Machado and Schirru (2002) is 48 after applying one-fourth core symmetry to the original 192. If one-eighth symmetry was applied, the number of channels would be 24.

It should be noted that various approximations were made to the reactor model so that optimisation calculations can be performed using the in-house code EVENT in a three-dimensional geometry. As will be seen from our optimisation studies, the 'excess' reactivity of the fresh core has been maximised as an illustrative example only. When the actual core was loaded, the CONSORT reactor went critical with $22\frac{3}{4}$ elements in the 1960s. In our model this was increased to 24 fuel elements with no control rods present in the core, in order to create a clean-core state for excess-reactivity optimisation.

7.2.2 LP Representation

A straightforward LP representation can be established using an integer vector. For example, an integer vector form of an LP is given below:

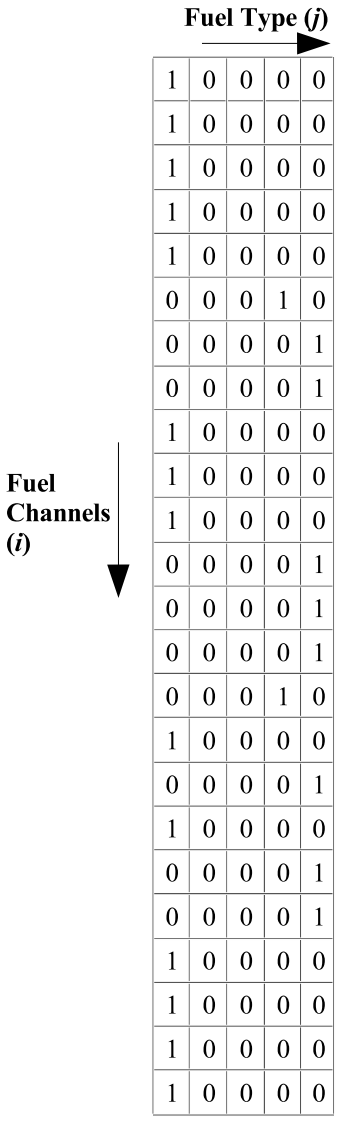
$$LP = [1, 1, 1, 1, 1, 4, 5, 5, 1, 1, 1, 5, 5, 5, 4, 1, 5, 1, 5, 5, 1, 1, 1, 1] \tag{7.1}$$

Each entry in this vector indicates a fuel type; in this case, MARK I_A is 1, MARK I_B is 2, MARK I_C is 3, MARK II is 4 and MARK III is 5, as shown in table 7.2. The index of this vector indicates the fuel channel index shown in figure 7.2.

It will be recalled from the encoding method in the previous chapter that an alternative representation is a binary matrix encoding, which is used here for a better understanding of the EDAs' concept. In practice, the binary matrix can be coded into an integer vector easily. The binary matrix of the LP shown in figure 7.3 is illustrated in figure 7.5.

7.2.3 Problem Model, Heuristic Information and Surrogate Model

Having defined an optimisation task (maximising the K_{eff}) and a properly encoded fuel inventory to be used to achieve it, the objective function needs to be implemented to carry out the K_{eff} calculation. In this project, we built a simulation



1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
0	0	0	1	0
0	0	0	0	1
0	0	0	0	1
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
0	0	0	0	1
0	0	0	0	1
0	0	0	0	1
0	0	0	1	0
1	0	0	0	0
0	0	0	0	1
1	0	0	0	0
0	0	0	0	1
0	0	0	0	1
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0

Figure 7.5: The binary matrix representation used to define LPs in the optimisation algorithms showing the example LP in figure 7.3

model for K_{eff} calculation using the finite element modelling software EVENT.

To speed up the calculation, a much faster surrogate modelling technique using Artificial Neural Networks (ANNs) was developed and eventually replaced the full model to estimate the objective function K_{eff}. The ANN model cannot be built without the finite element modelling.

Problem Model

Given the 'fresh core' model with 24 fuel elements and no control rods and aluminium clad, as well as the five basic types of fresh fuel elements, there is less perturbation during the neutron transport in the reactor core, which leads to a more regular solution space of loading patterns. This case represents the situation in which the fuel inventory has to be categorised manually into a few 'types', in order to reduce the search space or computational cost to a reasonable level.

The Heuristic Data

Using heuristic data in stochastic optimisation algorithms can be a good performance enhancement, in terms of the convergence speed and solution quality. For example, the distance between the cities in the TSPs has been used in a few optimisation algorithms, such as in the Ant Colony Optimisation (ACO), which can be regarded as a variant of the EDAs.

It will be recalled from the description of the use of heuristic data in the EDAs in the previous chapter that the construction of the heuristic data is based on how the LP is encoded. In this case, using the binary matrix representation of an LP, as shown in figure 7.5, it can be observed that the basic building-block of a candidate LP is a 'fuel type'-to-'fuel channel' assignment. In other words, an LP is constructed by combining possible building-blocks.

Therefore, useful heuristic data is the likelihood of these building-blocks being used in a promising candidate LP. For this case, we propose to issue a 'weight' for every possible fuel-type-to-fuel-channel assignment, which will be used in our EDAs

when searching for optimal LPs.

This information is called the 'stand-alone K_{eff} with fuel coupling' (stand-alone K_{eff} for short). It is inspired by the technique used for ANN training in Erdogan and Geckinli's work (Erdogan and Geckinli (2003)), derived from the coupled reactor theory (Avery (1958)), and then applied in our approach for reactor loading pattern-optimisation problems. The construction method and more information about the stand-alone K_{eff} can be found in Appendix A.

For this particular test case, the stand-alone K_{eff} is organised in a 24×5 matrix, shown in table 7.3. Each entry of the matrix indicates the contribution of a particular fuel element-channel assignment to the objective function - the overall K_{eff}.

Each entry (i, j) (i for channel and j for fuel type) is the calculated core K_{eff} when loading fuel j to channel i with all other channels filled with MARK I_A. This is why MARK I_A has an identical value for different channels. MARK I_A is used to form a generic core state because it has the median reactivity. Further discussion is presented in Appendix A.

Surrogate Modelling by Neural Network

To calculate the K_{eff} using EVENT requires a model file, a material property file and a simulation run, which can be very expensive in terms of computational time.

The main drive for using an ANN as a surrogate model of the EVENT model is the massive time saved in the evaluation of the K_{eff}s for candidate LPs. Given sufficient simulation results by EVENT, an ANN can be trained using the resulting 'LP-K_{eff}' pairs, and it can then be used as a fast K_{eff} predictor for any given LP.

We can simply treat the ANN as a 'black box', and therefore there is no direct dependence between the ANN and the optimisation algorithm. However, having the ANN provide extremely fast objective function evaluation is crucial in our optimisation algorithm research as well as for other stochastic algorithms, because they normally require huge numbers of objective function evaluations. Extensive

Channel No.	MARK I_A	MARK I_B	MARK I_C	MARK II	MARK III
1	1.20460	1.19972	1.19744	1.20690	1.20883
2	1.20460	1.19919	1.19654	1.20717	1.20930
3	1.20460	1.19919	1.19654	1.20717	1.20930
4	1.20460	1.20164	1.20016	1.20611	1.20735
5	1.20460	1.20164	1.20016	1.20611	1.20735
6	1.20460	1.18920	1.18184	1.21091	1.21600
7	1.20460	1.19186	1.18581	1.21004	1.21448
8	1.20460	1.19186	1.18581	1.21004	1.21448
9	1.20460	1.19833	1.19525	1.20754	1.20995
10	1.20460	1.19833	1.19525	1.20754	1.20995
11	1.20460	1.19652	1.19252	1.20831	1.21133
12	1.20460	1.19652	1.19252	1.20831	1.21133
13	1.20460	1.18736	1.17857	1.21153	1.21700
14	1.20460	1.18736	1.17857	1.21153	1.21700
15	1.20460	1.18168	1.16935	1.21312	1.21972
16	1.20460	1.19797	1.19469	1.20778	1.21038
17	1.20460	1.19797	1.19469	1.20778	1.21038
18	1.20460	1.19216	1.18589	1.20989	1.21412
19	1.20460	1.19216	1.18589	1.20989	1.21412
20	1.20460	1.19092	1.18435	1.21043	1.21515
21	1.20460	1.20156	1.20004	1.20615	1.20742
22	1.20460	1.20156	1.20004	1.20615	1.20742
23	1.20460	1.19907	1.19636	1.20728	1.20949
24	1.20460	1.19907	1.19636	1.20728	1.20949
$S.D.$	0.00000	0.00545	0.00827	0.00200	0.00356

Table 7.3: The calculated stand-alone K_{eff}s using EVENT for the five different fuel types in the CONSORT reactor test case 1. The $S.D.$ is the fuel channel-wise standard deviation.

LPs Set	Total No. LPs	Error < 0.5%	Error < 1%	Average Error
Training Set	767	762	767	0.12%
Unseen Set	308	290	307	0.19%

Table 7.4: The test result of the ANN predicting the K_{eff} for Test Case 1 - the fresh core

algorithm research is only feasible when this surrogate modelling technique is used.

In this test case, the ANN predicts the K_{eff} with an average error of 0.19% against the EVENT calculation, as shown in table 7.4. The speed gain is at least of a factor of two. When a fine mesh is used in EVENT, the speed gain can be much larger.

In summary, the surrogate modelling by ANNs is a robust and efficient method for predicting key parameters in reactor loading pattern optimisation applications. It was used extensively in this work in order to perform fast and accurate K_{eff} predictions. Additional details about the application of ANNs in reactor loading pattern optimisation are discussed in Appendix B and in one of the publications based on this work (Jiang et al. (2008)).

7.2.4 The Algorithms

Having defined the optimisation problem, built the computer simulation model, assembled useful heuristic information and formed a fast objective function evaluator (i.e. the ANN), we develop a couple of EDA-based algorithms as well as the benchmark GAs.

It will be recalled from the algorithms' description in the previous chapter that a basic EDA works as follows:

(1) initialise a population of LPs randomly;

(2) select some good LPs according to their K_{eff} or other objective function(s);

(3) build or update the probability model using the following:

$$P_{(t+1)} = (1 - \alpha) \cdot P_{(t)} + \alpha \cdot X \tag{7.2}$$

where P is the probability model for core channels; t is the number of the iteration or generation in EAs' terminology; P_0 is a uniform distribution model; α is a scalar between $[0, 1]$; and X is the statistical information extracted from the selected LPs;
(4) sample new LPs from the updated probability model;
(5) if stop criteria are not met, go back to (2), otherwise stop.

In EDAs, the building block for LPs is fuel type (or element) - channel assignment. When a population of candidate LPs is given, for each channel we record the probability that it is occupied by each fuel type. This statistical information plays a key role in EDAs, being used, perturbed and evolved during the whole optimisation process, as in Step 3 above.

EDAs use the probability model and a sampling operator to generate a new population of candidate solutions (step 4). The resulting LPs may not satisfy hard constraints, such as that the two MARK_II elements must be inserted into channels 6 and 14. The simplest way to deal with such LPs is to discard them.

The basic algorithm described above is named EDA_S (S for simplicity) based on the methodology stated in the previous chapter. For this particular application, we used two variants based on this basic approach.

An EDA with an Elitism Strategy

A modified algorithm is developed by combining EDA_S with a generic elitism strategy, called EDA_G (G for generic). It is identical to EDA_S except that an elitism strategy is used to improve the exploitation of the algorithm, and therefore fast convergence and better solution quality are expected.

One of the well-known problems of GAs (and also of other population-based optimisation algorithms) is that they often fail to find a local optimum, even when the current best solutions are very close to it. This can be caused by inappropriate

encoding, crossover method or random perturbations used in evolutionary operators (i.e. mutation).

An elitism strategy can be used as in Poon and Carter (1995) to improve the exploitation near the current best LP. Normally it is to ensure that the current best solution is kept at all times so that, hopefully, the crossover and mutation operators may generate more candidate solutions around the current best.

The same method can be employed in the EDAs without any difficulty. We use the idea but adapt it to fit into the EDA framework. The adapted elitism strategy is to perturb the probability model directly using the current best solution, as a result of which, the forthcoming search for promising LPs is explicitly biased in the direction indicated by the current best.

Since the LP representation and the probability model P have an identical data structure, the perturbation on P can be done by adding the weighted binary-matrix form of the current best LP to the sampling model P. A modified probability model updating method is given below:

$$P_{(t+1)} = (1 - \alpha) \cdot P_{(t)} + \alpha \cdot X + \eta \cdot X_b \tag{7.3}$$

In the additional term $\eta \cdot X_b$, X_b is the best solution found so far and η is a scalar. This term guides the search towards the direction of the current best solution. It will keep the search close to the current location if the best is near the centroid of the population; if the best is far away from the centroid, it will accelerate the movement of the population centroid.

By adding the elitism term in this unconstrained manner, the P is not normalised. Manual normalisation may be necessary. Alternatively, a constraint method can be used, as below:

$$P_{(t+1)} = \omega 1 \cdot P_{(t)} + \omega 2 \cdot X + \omega 3 \cdot X_b \tag{7.4}$$

where $\omega 1$, $\omega 2$ and $\omega 3$ are scalars and add up to 1. It is not different to the uncon-

strained method, in principle. In this work we choose the unconstrained method in order to keep the algorithm parameters less dependent on each other.

To summarise, in normal GAs that use a crossover operator to generate more solutions, the elitism strategy is always to select the current best individual, thereby increasing the survival chance of the better chromosome or building blocks in it. In EDAs, the selection pressure is applied directly to the probability model in a quantitative manner as described above, which makes it easier to control and allows further performance tuning.

An EDA Combined with Heuristic Information

The ideal application of the 'black box' optimisation methods, including GAs and EDAs, is for situations in which the search space is large, complex, non-convex, or poorly understood. However, for a particular problem, there is often background information or expert knowledge that can be utilised in order to refine the optimisation. Furthermore, proper use of heuristic information has been known as a good method to improve the stochastic optimisation algorithms.

An interesting example is the use of heuristic information in the Ant Colony Optimisation (ACO) (Dorigo and Gambardella (1997)) applied to the Travelling Salesman Problem (TSP). In ACO for TSP, the probability of choosing a path from the current city to the next one depends on how often this path was used by some known good tours (positive) and its length (negative). The path length has been shown to be crucial to finding near-optimal tours for TSPs quickly.

Another example in reactor loading pattern optimisation is the GA with HTBX for PWR loading pattern design (Poon (1992)), in which the reactor core structure and the reactivity of fuel elements are used to help encode LPs and generate new LPs. The HTBX encoding maps the search space to a reactivity distribution space, by using reactivity ranking to identify fuel elements and proposing new LPs by swapping physically adjacent sub-patterns between existing LPs.

Compared to breaking and recombining the sub-LPs represented by random

fuel elements IDs, an LP's reactivity distribution is strongly correlated with a reactor's key parameters. The experimental results showed that GA with HTBX accelerated the search and outperformed other GA-based algorithms (Poon (1992)).

Similar idea underlies these examples. It is to derive a correlation between the objective function and the building blocks of the problem model, by using the problem-dependent information, such as the distance between two cities in TSPs and the sub-patterns of reactivity distribution of the whole reactive core.

This correlation should be obvious, and the stronger it is, the better. It may not be direct, though. Given that the original reason for using stochastic algorithms is that the problem is likely to be poorly understood or the search space is huge and irregular, finding the proper heuristic information could be difficult.

We discovered a very useful form of heuristic information for our test case - the stand-alone K_{eff} with fuel coupling. It is a set of weights for each possible fuel element-channel assignment, which is the building-block of our problem model. It is known that there is a strong correlation between the stand-alone K_{eff} and the overall core K_{eff} based on the coupled reactor theory (Avery (1958)). More information is available in Appendices A and B.

In EDAs, the probability model represents the distribution of the promising solutions, or it can be regarded as the centroid of the current population. This model can be biased, or pushed, in a given direction indicated by a piece of heuristic information.

EDA_H, which uses this heuristic information, is identical to EDA_G except that EDA_H samples a new population of LPs using the updated probability model P together with some heuristic information. The methods by which heuristic information is used in EDAs are described in the previous chapter. The sampling probability is:

$$P' = P \cdot H^{\beta} \tag{7.5}$$

where H contains heuristic information - the stand-alone K_{eff} - and β is an expo-

nential scalar adjusting the weight between population-based learning and heuristic information. H has an identical data structure to P. For this problem, we use the stand-alone K_{eff} with fuel coupling as H. The entry $H(i, j)$ is the calculated stand-alone K_{eff} of loading fuel j to channel i and other channels filled with MARK I_A. A larger $H(i, j)$ increases the probability of loading fuel j to channel i. An example is given in figure 6.2.

Similarly, in EDA_H, the probability of assigning a fuel type j to a fuel channel i depends on how often this assignment appeared in some known good LPs (positive) and its corresponding stand-alone K_{eff} with fuel coupling (positive), which can be considered a measurement of the 'contribution' that a certain fuel element makes to the objective function, the overall K_{eff}, when it is inserted into a specified channel in a generic reactor environment. The calculated stand-alone K_{eff} value for this test case is given in table 7.3.

To demonstrate how H affects the optimisation, let us assume that there are 10 candidate LPs, and five of them use a MARK III fuel element at channel 13. The other five LPs use MARK I_A at channel 13. After the application of heuristic information with $\beta = 4$, the probability of using a MARK III at channel 13 is higher:

$$P^h(13, MARKIII) = 0.5 \cdot 1.217^4 \approx 1.09681 \tag{7.6}$$

while

$$P^h(13, MARKI\_A) = 0.5 \cdot 1.2046^4 \approx 1.05279 \tag{7.7}$$

where 1.217 and 1.2046 are the respective stand-alone K_{eff}s. From this example we can see that in a case in which the population-based learning cannot give clear instructions, the heuristic information will lead the search. Note that P^h for each channel should be normalised after being multiplied by H.

Also, from the stand-alone K_{eff} with fuel coupling table, $H(13, 5) = 1.217$ is larger than $H(22, 5)$ which is 1.20742. This means that loading MARK III into

channel 13 is potentially more effective than loading it into channel 22. Even though no channel dependence is considered in the EDAs presented in this work, the use of heuristic information in a 'weight-like' form $P \cdot H^{\beta}$, can compensate this by applying the 'weight' containing the position-wise and adjacent fuel information. For channel 13, the vectors that give information about the relative probabilities for each of the five fuel types, j, are

$$P(13, j) = [0.2, 0.2, 0.2, 0.2, 0.2] \tag{7.8}$$

$$H^4(13, j) = [2.10557, 1.9876, 1.93057, 2.15445, 2.19362], \tag{7.9}$$

After normalisation, P^h is:

$$P^h(13, j) = [0.2030, 0.1906, 0.1816, 0.2077, 0.2115], j = MARKI\_A, ..., MARKIII \tag{7.10}$$

Similarly,

$$P^h(22, j) = [0.2004, 0.1984, 0.1974, 0.2015, 0.2023], \tag{7.11}$$

It can be seen that the assignment of MARK III to channel 13 is more likely than its assignment to channel 22.

Although we can expect fast convergence by using heuristic information in evolutionary algorithms, local information like stand-alone K_{eff} with fuel coupling is not always available and highly problem-dependent. So the calculation/extraction of the heuristics is problem-dependent, but the way of using it in EDAs, in the form described above, can be generalised.

In this section, we described the two proposed EDAs, the EDA_G and EDA_H. EDA_G is a pure 'black box' method that does not require any problem-dependent information, but it does use a generic elitism strategy to improve its performance. EDA_H makes use of the stand-alone K_{eff} with fuel coupling to improve the optimisation.

Benchmark Genetic Algorithms

GAs are efficient and versatile algorithms for tackling complex, large-scale combinatorial optimisation problems in the sciences and engineering. There are many research works that have contributed to the application of GAs for reactor fuel management optimisation problems. For this reason, we are going to use a GA as a benchmark.

Our GA-based algorithm is described below:

(1) initialise the LP population randomly;
(2) select some LPs according to their K_{eff};
(3) generate the new population of LPs by applying crossover and mutation to the selected individuals;
(4) if stop condition is not met, go back to step 2, otherwise stop.

One of the key elements to any GA's success is its crossover operator. A study of GA crossover operators for ordering applications can be found in Poon and Carter (1995). For this test case, we implemented the following crossover operators for GAs: the 2-Point Crossover (2PX) with 1-D integer vector encoding, and the Heuristic Tie Breaking Crossover HTBX Poon (1992) with 2D permutation representation.

The reason for which we chose 2PX is because it is classic and generic. It is one of the first crossover operators used in GAs and has been widely applied to various applications. Unlike the 2PX, the HTBX was specifically designed for reactor loading pattern optimisation. It is based on a study of different crossover operators for ordering problems and used problem-dependent information to improve its performance.

According to the different crossover operators, these GAs are referred to as follows:

(1) GA_2PX: A GA with 2PX (refer to Section 3.3 and Appendix C)

(2) GA_HTBX: A GA with HTBX (Section 3.3)

Due to the difference between 2PX and HTBX, different encoding methods were used. For 2PX, the integer-vector form of LP representation is used, just as in the EDAs; for HTBX, a permutation representation is used. The fuel element ID is defined by reactivity ranking.

7.2.5 Results

The EDAs developed were tested extensively and compared against GAs that employed different crossover operators. The K_{eff}s are calculated by using a trained ANN instead of EVENT simulation. By doing so, we are able to run more LP evaluations in reasonable time.

A proportional selection method on ranked fitness (Baker (1985)) value rather than the raw value of K_{eff} is used for all EDAs and GAs. It should be noted that this method is statistically equivalent to a two-person tournament selection scheme. In practice, the latter is more computationally efficient (Whitley (1989)).

Because different encoding methods are used, the mutation operators in the EDAs and GAs are different too. In our EDAs, it is to randomly choose a fuel element and then assign it to a fuel channel that is also randomly chosen. For GAs, the mutation operator is the Swap mutation in Oliver et al. (1987), which is to randomly swap the fuel elements at two different channels.

There are other selection and mutation methods. However, the ones we chose generally perform better, and research on different selection or mutation operators is beyond the scope of this work and will not be investigated further.

All algorithms use a normal 'elitism' strategy, which means the best solution so far is guaranteed to be passed to the next generation. In EDA_G and EDA_H, the modified 'elitism' strategy is also switched on, i.e. the sampling probability model is perturbed directly by the current best.

An initial population consisting of fifty LPs is randomly generated and used

Algorithms	Population Size	Max Generations	α	β	η	Mutation Rate	Crossover Rate
EDA_S	50	2000	0.001	N/A	N/A	0.05	N/A
EDA_G	50	2000	0.001	N/A	0.01	0.05	N/A
EDA_H	50	2000	0.001	30	0.01	0.05	N/A
GAs	50	2000	N/A	N/A	N/A	0.05	0.9

Table 7.5: Parameter settings for the numerical experiments giving the population size, total number of generations, mutation and crossover rates.

Algorithms	Best	Average	Standard Deviation
EDA_S	1.20173	1.19900	0.00108
EDA_G	1.20578	1.20572	0.00004
EDA_H	1.20578	1.20575	0.00005
GA_2PX	1.20573	1.20520	0.00041
GA_HTBX	1.20578	1.20537	0.00033

Table 7.6: The maximum K_{eff} found by EDAs and GAs from 30 independent runs and their corresponding average and standard deviation.

as the same starting point for all the algorithms. All experiments are based on 30 independently created runs. Only one initial population has been generated and it was used for all the runs of all the algorithms.

It should be noted that the performance of GAs and other evolutionary algorithms can be very sensitive to the values of their control parameters. We tuned the parameters used in the GAs and EDAs carefully to ensure the validity of the comparison. The well-tuned parameters are summarised in table 7.5. Also, an empirical study of parameter sensitivities in the developed EDAs is presented in section 7.5.

We recorded the best solution found in each generation, and plotted their average values from 30 independent runs. This is shown in figure 7.6. Experimental results after 100,000 LP evaluations are given in table 7.6. The maximum and minimum objective function values (error bounds) found among 30 independent runs of EDA_H and GA_HTBX are recorded and plotted in figure 7.7.

The results show that EDA_H, EDA_G and GA_HTBX find the same 'optimal' solution:

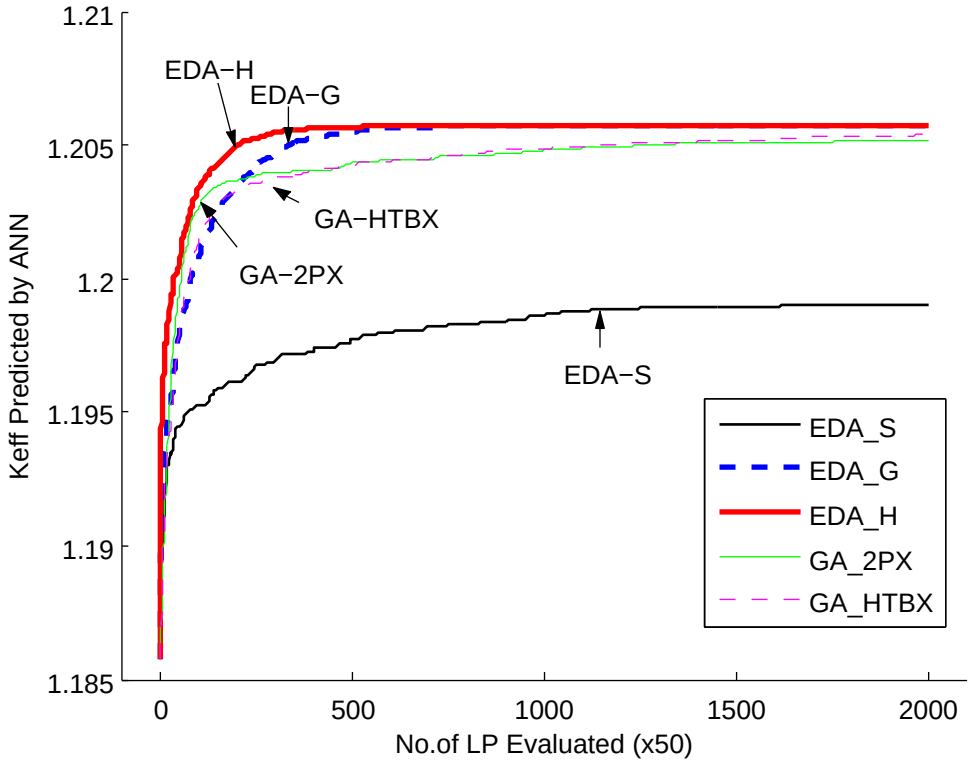


Figure 7.6: Results showing the averaged optimisation process of EDAs and GAs on the CONSORT reactor case

$$LP_{OPT} = 1, 1, 1, 1, 1, 4, 5, 5, 1, 1, 5, 1, 5, 5, 4, 1, 1, 5, 5, 5, 1, 1, 1, 1 \tag{7.12}$$

$$K_{eff} = 1.20578 \qquad \text{(Predicted by ANN)} \tag{7.13}$$

Both EDA_G and EDA_H outperform the tested GAs in optimising the K_{eff}s. It was also found that the EDAs have smaller variations compared to GAs, which can be seen by the average, standard deviation and error-bound values from the 30 independent runs.

The better GA of the two GAs is the GA_HTBX, with a similar performance close to EDA_G. GA_HTBX and GA_2PX converge slightly faster than EDA_G initially but EDA_G outperforms them later. This is a well-known problem with GAs. They are efficient in finding a local neighbourhood containing a good solution,

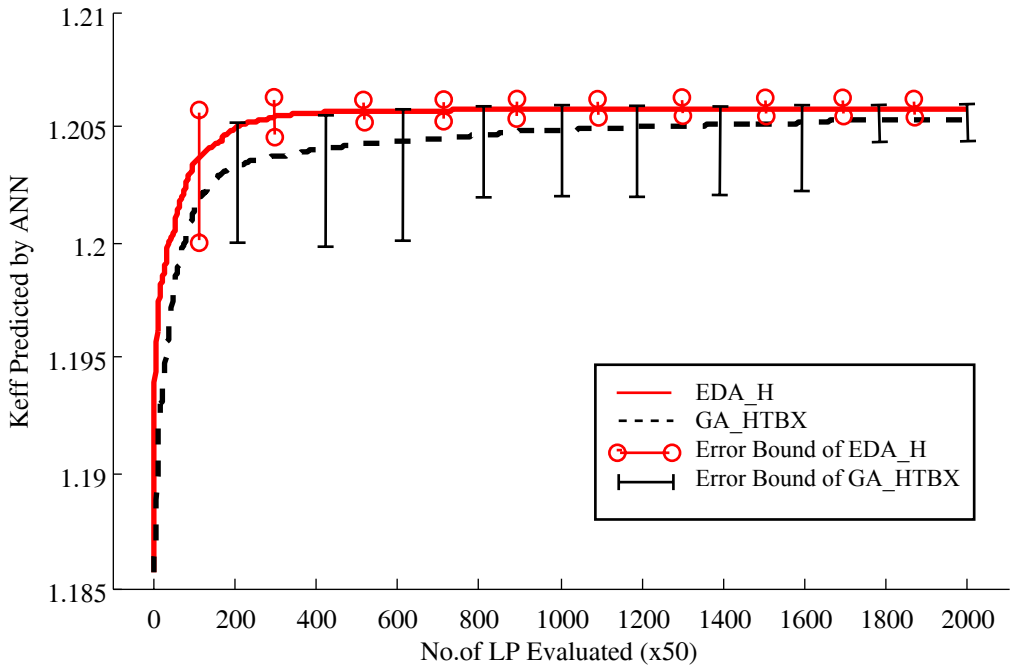


Figure 7.7: Results showing the averaged optimisation process of EDAs and GAs on the CONSORT reactor case with error bounds

but they are poor local optimisers.

It is not surprising that the GA_2PX performed very well and only slightly worse than HTBX at the end of the search. Considering the fuel inventory in this case, the integer vector form of LP representation in GA_2PX is a more natural and efficient encoding than the permutation form in GA_HTBX. There are in total 74 fuel elements in the hypothetical fuel store, and only five different fuel types - which means that, in the permutation form of an LP, many of the entries are simply identical fuels that have been given a random ID. When a crossover or mutation operator is being executed, it could be swapping identical fuel elements. That is why GA_2PX with the integer vector LP encoding converges quickly at the beginning. This is another good example of a typical GA - good at locating a good area very quickly, but inefficient in converging to a local optimum.

EDA_G and EDA_H outperform the intelligent crossover operator HTBX in terms of convergence speed and stability. EDA_G uses neither physical information about fuel assemblies nor the reactor core's 2D structure, but does use a new 'elitism-

guided' term to make full use of the best known LP so far. The effect of this is to improve the exploitation near the current best and therefore a better local convergence is achieved.

EDA_H makes use of newly developed problem-specific information, the stand-alone K_{eff} with fuel coupling, which includes the consideration of the channel position of a fuel element and the adjacent fuel elements, to improve the optimisation. It does not consider the core structure explicitly, but employs the stand-alone K_{eff} with fuel coupling to help the optimisation. This information contains the physical character of a fuel assembly, its position (fuel channel) and the neighbourhood fuel coupling. So the stand-alone K_{eff} with fuel coupling is not only helpful in speeding up the convergence, but also enables the EDAs to use the reactor core channels' position and neighbourhood fuel dependence information.

In addition, the permutation representation used in GA_HTBX, as discussed earlier in this section, has to include the whole 'fuel store' including in-core and out-core fuel assemblies to ensure that the entire search space is explored, which greatly increases the problem size and computational time, and may cause slow convergence and local-minimum trapping problems due to the inefficient crossover and mutation caused by the encoding. The EDA approach does not need the out-of-core fuel store information, because the probabilities sampling operator allows it to search the entire search space.

To summarise, we set up a K_{eff} optimisation problem from a fresh-core state of a realistic reactor and tested the algorithms EDA_S, EDA_G and EDA_H on it. The results are compared to the benchmark GA_HTBX. We use this test case to demonstrate how the algorithms work and to test the algorithms' performance on a relatively simple case - with a fresh-core state, no control rods and an encoding method that simplifies the fuel store.

The EDA_G and EDA_H used the new 'elitism-guided' probability updating method and problem-dependent information (only EDA_H) to improve local convergence. The conclusion from this numerical experiment is that the proposed EDAs

are better, or as good as, the previously published GAs for this test case in terms of solution quality, convergence speed and stability.

It also needs to be noted that the LP encoding method does affect the optimisation considerably. It is clearly observed from the conventional GA_2PX, which performs only slightly worse at later stages of the search compared to the best tested algorithms. GA_HTBX employed heuristic information, including the fuel reactivity and core structure, but its permutation-based LP encoding method affected its performance considerably.

In addition, the proper use of the heuristic information helped the EDA_H outperform other algorithms due to its natural and efficient encoding, which strengthens the correlation between the building-blocks and the objective function, and the use of the stand-alone K_{eff}, which quantified this correlation based on the pre-calculation in a simulated generic core state.

7.3 Test Case 2: A Realistic Core Model

In Test Case 1, EDAs are tested under the assumption that a large number of fuel elements are available, and that some of them have a very small variance, which can be ignored. This is normally the case when the power plant is large. However, when the fuel inventory is relatively small or a high-accuracy LP solution is required, all fuel elements must be regarded as different to each other.

The core model (and the fuel inventory) is also simplified in Test Case 1 to form a start-up test. Due to the fact that there is no control rod or metal clad, and that only five different fuel types are available, the search space of Test Case 1 is relatively regular and simple.

Test Case 2 is designed to be a more complicated case. Firstly, it requires higher accuracy, so each fuel element will be encoded explicitly according to its reactivity, the K_{∞}. Also, the core state is more realistic, with two control rods inserted, and with the aluminium clad as part of the structure. The search space is larger and more irregular, due to more reactivity variance among fuel elements and

the perturbation from the control rods and metal clad.

We demonstrate how the EDAs are adapted to this class of problems, and the results are compared to the well-established GA_HTBX. Based on the results from Test Case 1, only EDA_G, EDA_H and GA_HTBX are chosen. EDA_S was important for understanding the probability model sampling idea, but not ideal for solving realistic loading pattern optimisation problems. GA_2PX is not used, as it will produce infeasible LPs in their permutation representation, as a result of which special treatment will be needed to repair them. As pointed out in Poon's work (Poon (1992)), GA_HTBX is one of the best candidates for this particular application.

7.3.1 The Optimisation Task and the Fuel Inventory

A new core state representing the reactivity of the fuel elements, approximately extrapolated to the present time (2006), was constructed on the basis of fissile mass estimations. The reactor physics code WIMS8A was used to obtain the multi-group constants that are then used in the 3D homogenised EVENT computational model. By doing so, each fuel element contains a different amount of fissile material and is therefore essentially different in reactivity compared to all others.

The optimisation task is to find the optimal loading pattern that maximises the overall K_{eff} for this modified CONSORT core, given that the fuel inventory consists of 35 fuel elements, as shown in table 7.7. In this case, each fuel element will be different and will have a unique ID. All the fuel elements are ranked by their infinite multiplication factor, K_{∞} (or reactivity), and the ranking number is then used as their IDs.

The core structure is the same as in Test Case 1, except that control rods 1 and 2 are fully inserted. Control rods 3 and 4 are not included in the EVENT model, since they act as shut-down rods to ensure safety (see figure 7.2).

The hard constraint remains the same, in that the two MARK II type fuel elements (fuel elements 10 and 11 in this case) must always be inserted into channels

Ranking	1	2	3	4	5	6	7
K_∞	1.68	1.68	1.68	1.68	1.67	1.67	1.67
Ranking	8	9	10	11	12	13	14
K_∞	1.67	1.67	1.61	1.60	1.54	1.54	1.53
Ranking	15	16	17	18	19	20	21
K_∞	1.53	1.53	1.53	1.53	1.53	1.52	1.52
Ranking	22	23	24	25	26	27	28
K_∞	1.52	1.52	1.52	1.52	1.52	1.52	1.52
Ranking	29	30	31	32	33	34	35
K_∞	1.52	1.52	1.52	1.52	1.51	1.13	0.77

Table 7.7: The CONSORT reactor fuel store information of the modified Test Case 2

6 and 15, due to safety and operational constraints. Given the core plan, the fuel inventory and the constraints, the whole search space is $P_{33}^{22} \approx 2.18 \times 10^{29}$.

7.3.2 LP Representation

Since each fuel element needs to be encoded explicitly, we use a permutation to represent the whole fuel store for this test case. The K_∞ ranking numbers are used as their unique IDs. An LP is then represented as an integer vector containing a full permutation of integers in $[1, 35]$ (35 fuel elements). The first twenty-four integers indicate the in-core loading pattern, and the rest of them are not used in this loading pattern. An example is given below.

$$LP = [1, 2, 3, 4, 5, 10, 6, 7, 8, 9, 12, 13, 14, 15, 11, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35] \tag{7.14}$$

It should be noted that in this case, the order of out-of-core fuel elements is not relevant to the in-core LP. However, for the benchmark GA_HTBX to be able to explore the whole search space, the full permutation is required. This is because the crossover operator needs the LPs to contain all the information of the fuel inventory, so that it can regenerate valid new LPs.

In the EDAs, the LP representation is similar to the one used in Test Case 1. It is coded in an integer vector in the programs, but is best explained as a binary matrix form similar to the one shown in figure 7.5. The LPs can be represented only by the first 24 entries without losing any necessary information. So an encoded LP is an integer vector with 24 entries. The corresponding binary matrix form is a 24×35 matrix. There is only one '1' in each row and each column, because each fuel element can only be used once in one fuel channel. The same 24×35 matrix structure is used for the probability model (but with real values). By using it, all of the fuel elements will be considered when filling a fuel channel, no information will be lost and the LP representation is more compact.

7.3.3 Problem Model, Heuristic Information and Surrogate Model

Problem Model

Based on the modified test case, a 3D finite element model was built using EVENT. Unlike Test Case 1, the fuel inventory was recalculated and two control rods are fully inserted. The aluminium clad is also included in this model, as shown in figure 7.4.

This model represents a much more realistic core state. It is more complicated, and more noise and uncertainty are expected, due to the presence of the control rods and metal clad, as well as the variance among the fuel elements.

The Heuristic Data

To generate the stand-alone K_{eff}, we insert a fuel element j in channel i, filling all other channels with a 'generic' fuel m. An LP is then created and examined by the simulation software EVENT to obtain its K_{eff}. This result is recorded as the stand-alone K_{eff} with fuel-coupling information for fuel j in channel i. This calculation is repeated for all the fuel elements and all channels, and the results can

be presented in a 24×35 matrix. Each entry $[i, j]$ of this matrix represents the 'spatial contribution' of assigning fuel j to channel i in a more realistic context.

This set of data is too large to be shown in the main text, but it is available in Appendix A. The use of the stand-alone K_{eff} remains the same as in Test Case 1, as given in the following equation:

$$P^{'} = P \cdot H^{\beta} \tag{7.15}$$

where the $P^{'}$ is the probability model used to sample new candidate solutions, P is the probability model updated by the EDA algorithm, H is the stand-alone K_{eff} and β is an exponential scalar. The data structures of the probability model P and H are identical to each other as well as to the binary matrix form of an LP.

The stand-alone K_{eff} with fuel coupling can be used directly as H. If the variance of different assignments of fuel elements to channels is too small even when a large β has been used, our suggestion is to use the ranked stand-alone K_{eff} matrix instead of the original raw data. The total $24 \times 35 = 840$ entries in the original stand-alone K_{eff} matrix are ranked from 1 to 840, which represents the relative contribution of all the possible assignments of fuel elements to fuel channels. Unacceptable assignments (e.g. assigning fuel element 10 to any channel other than channel 5) should not be included in this ranking as it will disturb the effect of the stand-alone K_{eff}.

Surrogate Modelling by Neural Network

To accelerate the calculation of the overall K_{eff}, we used the neural network technique to build a fast surrogate model based on the EVENT simulation results. Given the pre-calculated candidate LPs and the resulting the K_{eff}s, the neural network can be trained as a fast and accurate K_{eff} predictor.

More details regarding the use of ANNs in this work can be found in the ANN section in Test Case 1, and in Appendix B. The ANN used for this test case was tested and the results are summarised in table 7.8.

LPs Set	Total No. LPs	Error $< 0.5\%$	Error $< 1\%$	Average Error
Training Set	1620	1611	1620	0.13%
Unseen Set	203	203	203	0.13%

Table 7.8: The test result of the ANN predicting the K_{eff} in Test Case 2: a realistic core state. The training set is the set of LP-K_{eff} pairs used to train the neural network, while the unseen set is the set of pairs that is not used in training.

Algorithms	Population Size	Max Generations	α	β	η	Mutation Rate	Crossover Rate
EDA_G	50	2000	0.001	N/A	0.01	0.05	N/A
EDA_H	50	2000	0.001	2	0.01	0.05	N/A
GA_HTBX	50	2000	N/A	N/A	N/A	0.05	0.9

Table 7.9: The well-tuned parameters settings used in EDAs and GAs for Test Case 2

7.3.4 Results

The parameters used are summarised in table 7.9. It should be noted that the β value in EDA_H is smaller in this case because we used the ranked stand-alone K_{eff} instead of the raw value calculated by EVENT model. The reason for this is that the variance among fuel elements is much smaller compared to Test Case 1, and it can be easily lost in the numerical calculations. Hence the β may have to be very large.

We prefer to keep β to reasonable magnitudes, e.g. an O(1) number. For this reason we suggested ranking all the entries in the stand-alone K_{eff} table. The ranking stretches the differences between the fuel element-channel assignments so that a much smaller range of β can be used. In this case, $\beta = 2$ is sufficient.

Numerical results after 100,000 LP evaluations are given in table 7.10. For all EDAs and GAs, the best solution found in each generation was recorded, and their average values from 30 independent runs are illustrated in figure 7.8. Figure 7.9 shows the maximum and minimum objective function values (error bounds) found among 30 independent runs of EDA_H and GA_HTBX. It is found that EDA_G and EDA_H algorithms both found better solutions than GA_HTBX, as well as better averaged best solutions found over 30 independent runs. The standard deviations

Algorithms	Best	Average	Standard Deviation
EDA_G	1.007480	1.007459	0.000039
EDA_H	1.007480	1.007477	0.000016
GA_HTBX	1.007400	1.007151	0.000221

Table 7.10: The maximum K_{eff} found by EDAs and GAs from 30 independent runs and their corresponding average and standard deviation

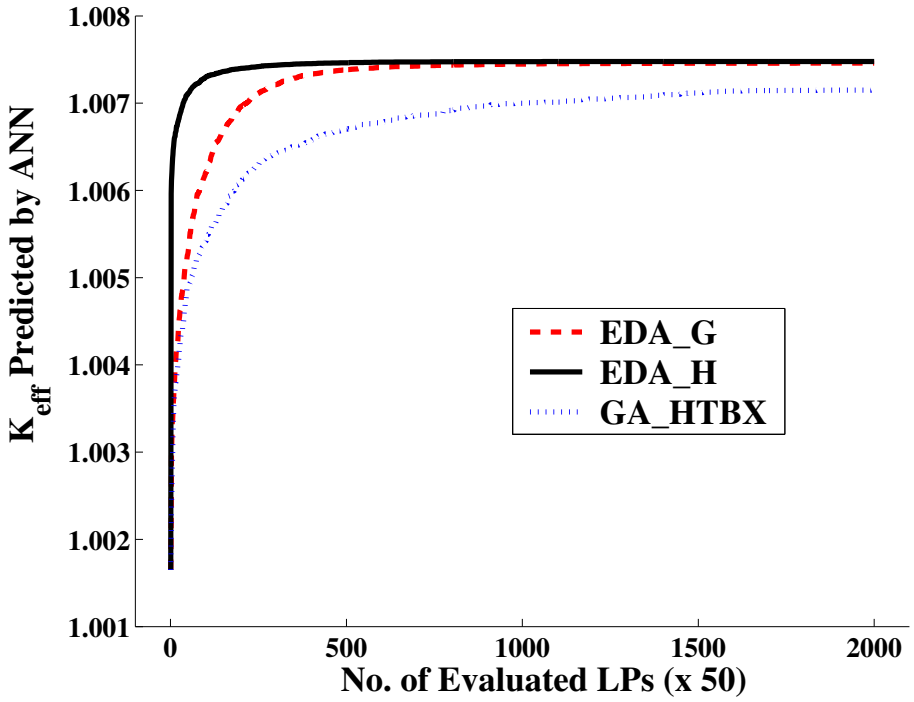


Figure 7.8: The averaged performance of EDAs and GAs against number of LP evaluations of the modified CONSORT case

also suggest that both EDAs converge faster than GA_HTBX.

To summarise, we set up a realistic core model based on the CONSORT reactor with the presence of control rods, metal clad and a much more accurate fuel inventory. This test case requires higher accuracy so that in the problem model, the fuel elements have their unique IDs. The experimental results show that EDA_H and EDA_G outperformed the benchmark GA_HTBX in terms of solution quality, convergence speed and stability.

The encoding method for LP representation is again very important in this test case. Neither EDA uses the core structure, but EDA_H does use the standalone K_{eff}, which contains position-wise information of the core channels implicitly.

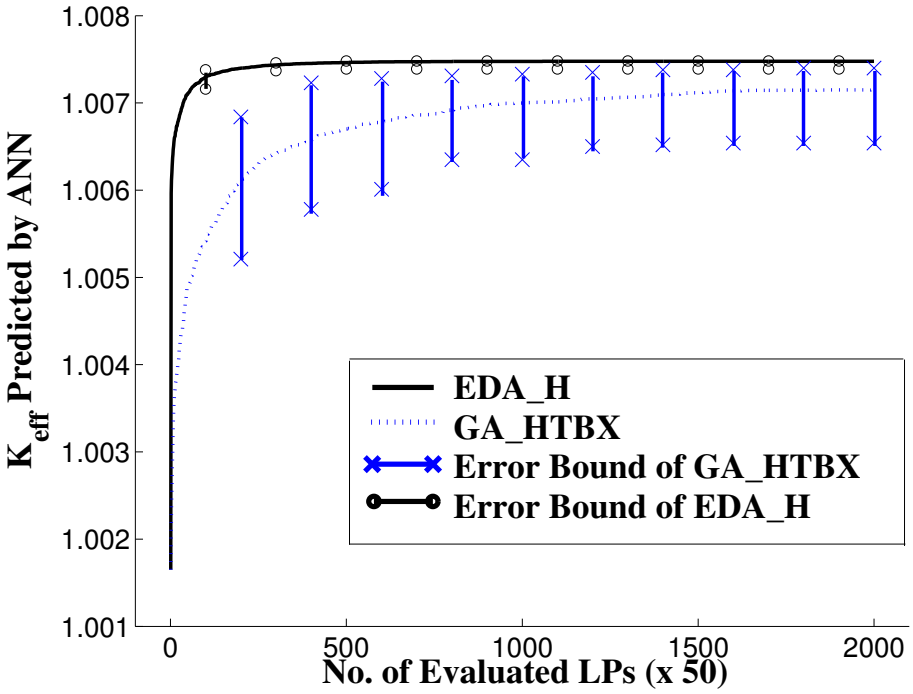


Figure 7.9: The comparison between EDA_H and GA_HTBX on the modified CONSORT case with error bounds

GA_HTBX has the core structure explicitly encoded in its candidate LP representations (Poon (1992)). However, the tie-breaking algorithms used in GA_HTBX somehow introduce random noise into the evolutionary process, which slows down the convergence. From previous GA research and experience in applications, it would be expected that GA_HTBX will find similar results to EDA_G and EDA_H if given more time, but there is no theory to prove this.

That said, the fast convergence of EDA_G and EDA_H may cause a premature problem, as they can become stuck at some local optima. GA_HTBX, due to more randomised noise being used, suffers less from the local convergence problem. In order to resolve this for EDAs, parameter tuning is necessary.

In addition, the use of the stand-alone K_{eff} in EDA_H is slightly different, as the ranked value replaced the raw value. The ranking stretches the differences between the basic building blocks of an LP (the fuel element-channel assignment), particularly when they are very small, and therefore ensures that the necessary information is kept and the magnitude of the free parameter β is reduced.

7.4 Test Case 3: A Realistic Core with a Constraint

Test Case 3 is designed to be an even more complicated case based on the core model used in Test Case 2. A power peaking constraint was added to the previous objective function. The algorithms to be tested are EDA_G, EDA_H and GA_HTBX. The main purpose of this section is to demonstrate the application of EDAs to LP optimisation with a typical constraint, and to observe their performance. Parameter sensitivity studies and preliminary experiments for multi-objective optimisation are also presented.

Because the CONSORT reactor is a low-power research reactor, it does not generate excessive energy in the core. In reality, the Power Peaking Factor (PPF) is not a realistic safety concern. That is why maximisation of K_{eff} without other constraints was set up as part of the CONSORT reactor 'Future Options' project (Ziver (2005)). We have changed the fuel reactivity to create this test case but the variation of the PPF is relatively small. It is introduced as an extra 'constraint' mainly for algorithm study.

7.4.1 The Optimisation Task and the Fuel Inventory

In this section, we use the PPF as a penalty term in the objective function as it is a practical way of dealing with constraints. The experiments are performed with a simple weighted sum of the K_{eff} and PPF. Constraints and multi-objective issues are discussed in a subsequent section.

The optimisation task is then to maximise an objective function that consists of the core K_{eff} and the PPF constraint as a penalty term, which is:

$$F = w1 \cdot K_{eff} + w2 \cdot PPF \tag{7.16}$$

where $w1$ and $w2$ are scalars adjusting the weights between the objective term and

Ranking	1	2	3	4	5	6	7
K_∞	1.80	1.79	1.78	1.77	1.76	1.75	1.74
Ranking	8	9	10	11	12	13	14
K_∞	1.73	1.72	1.71	1.70	1.69	1.68	1.67
Ranking	15	16	17	18	19	20	21
K_∞	1.66	1.65	1.64	1.63	1.62	1.61	1.60
Ranking	22	23	24	25	26	27	28
K_∞	1.59	1.58	1.57	1.56	1.55	1.54	1.53
Ranking	29	30	31	32	33	34	35
K_∞	1.52	1.51	1.50	1.49	1.48	1.47	1.46

Table 7.11: The modified CONSORT reactor fuel inventory for Test Case 3

the constraint term and PPF is the estimated power peaking factor given a loaded core. In this work, we used neutron flux density (thermal group) to replace the actual power in order to simplify the PPF calculation. It is calculated using the equation below:

$$PPF = \frac{\text{The Maximum Thermal Flux Density of all Fuel Channels}}{\text{Averaged Thermal Flux Density of all the Fuel Channels}} \tag{7.17}$$

It should be noted that in this test case, maximising F may not yield the optimal K_{eff} given the PPF constraint, because the two weights, $w1$ and $w2$, have a significant impact on the direction of the optimisation. A larger weight will bias the search towards the corresponding objective/constraint. In this case, $w1$ will be set to a positive value, e.g. 0.8, while $w2$ will be a negative value, e.g. -0.2. In practice, these two parameters should be adjusted to suit real-world applications.

The core structure and the hard constraint are both the same, as in Test Case 2. The size of the search space remains the same, and is approximately 2.18×10^{29}. However, in order to distinguish different fuel elements better, the fuel inventory has been modified to increase the reactivity difference between different fuel elements, which is summarised in table 7.11. The problem caused by spreading the reactivity evenly is that the variation of the PPF will be relatively small. In fact, in the CONSORT reactor, PPF has never been a real safety constraint because of its low power. The PPF is introduced for algorithm study only.

7.4.2 LP Representation

In this test case an LP is represented as a permutation vector, which is the same as in Test Case 2. (Refer to section 7.3.2.)

7.4.3 Problem Model, Heuristic Information and Surrogate Model

Problem Model

We used the same 3D finite element model used in Test Case 2, except for a different fuel inventory, summarised in table 7.11.

Heuristic Information

Similarly to the previous test cases, the heuristic information will be an estimated contribution of the building blocks (the assignments of fuel elements to fuel channels) to the objective function. Since there are two components in the objective function, the heuristic information for this test case will need to contain the information of both K_{eff} and PPF. We calculate one set of the stand-alone K_{eff}, which is a 24×35 matrix, as in the previous test cases. In addition, another matrix with an identical structure to the stand-alone K_{eff} is built, which can be called the stand-alone PPF.

To calculate the stand-alone PPF, an LP with a certain fuel element, j, inserted into one of the 24 fuel channels, i, and other fuel channels filled with a generic fuel elements, 'm', is set up and fed to an EVENT simulation. The estimated stand-alone PPF of this LP can be calculated using equation 7.18, which is different to equation 7.17, as this stand-alone PPF is intended to be a measure of the power generated by fuel element j in fuel channel i compared to the averaged power of all the fuel channels. Repeating this process for all the fuel elements and all fuel channels, a 24×35 matrix can be filled, just like the stand-alone K_{eff} is built. Both the stand-alone K_{eff} and PPF are shown in Appendix A.

For K_{eff} Prediction				
LPs Set	Total No. LPs	Error $< 0.5\%$	Error $< 1\%$	Average Error
Training Set	1560	1558	1560	0.08%
Unseen Set	195	195	195	0.08%
For PPF Prediction				
LPs Set	Total No. LPs	Error $< 0.5\%$	Error $< 1\%$	Average Error
Training Set	1560	1433	1558	0.23%
Unseen Set	196	181	196	0.23%

Table 7.12: The test results of the ANNs predicting the K_{eff} and PPF for Test Case 3

$$PPF = \frac{\text{The Thermal Flux Density of the Chosen Fuel Channel } i}{\text{Averaged Thermal Flux Density of all the Fuel Channels}} \tag{7.18}$$

The stand-alone K_{eff} and PPF are combined in the same way as they are used in the objective function:

$$H = w1 \cdot K_{eff} + w2 \cdot PPF \tag{7.19}$$

where H is the heuristic information and $w1$ and $w2$ are weight scalars. In this case, they will be set to exactly the same values as in the objective functions, i.e. $w1 = 0.8$ and $w2 = -0.2$. This H matrix contains the 'spatial' contribution of each fuel element when it is assigned to a specific fuel channel.

Surrogate Modelling by Neural Networks

Neural networks were developed to predict K_{eff} in the previous cases. In this case, two neural networks are used. One of them is for predicting K_{eff}, while the other one predicts the estimated PPF used in the objective function. The prediction test results of these two ANNs are summarised in table 7.12. The predictions from both ANNs are added together with weights $w1$ and $w2$ to evaluate the objective function value of a given LP.

7.4.4 Results

EDA_G, EDA_H and GA_HTBX are tested on this test case. All the algorithms were tested 30 times independently. In the previous test cases, all the simulations started with the same initial population. In this test case, all the algorithms generate an initial population randomly on the fly at the beginning of each independent run. This is to ensure a better exploration for all the tested algorithms.

We used the same parameter settings as in Test Case 2, summarised in table 7.9, because the core model is the same and only the fuel inventory is changed. It is generally known that evolutionary algorithms can be sensitive to their control parameters. For this reason, a study of parameter sensitivity will be presented in the next section.

The objective function changed significantly in this test case, because of the inclusion of the PPF term and the introduction of the two weights $w1$ and $w2$. To ensure a better understanding of this test case, we first tested the algorithms with only one of the two terms enabled in the objective function.

The first scenario is to enable the K_{eff} term only, which is done by setting $w1$ to 1 and $w2$ to 0. This test case then turns into a K_{eff} maximisation problem identical to the previous test cases. The heuristic information is also modified to exclude the stand-alone PPF. The results are summarised in table 7.13.

The results of maximising K_{eff} show a very similar pattern compared to the previous test cases. EDAs find better results than the tested GA. In addition, the standard deviations of 30 independent EDA runs are much smaller than GA_HTBX. The averaged performances from 30 independent runs are plotted in figure 7.10.

The second scenario is to enable the PPF term only, which is done by setting $w1$ to 0 and $w2$ to -1. This test case then turns into a PPF minimisation problem. The heuristic information is also modified to exclude the stand-alone K_{eff}. The test results are summarised in table 7.13.

The results of minimising PPF only for Test Case 3 show that both EDAs find smaller PPF values than the tested GA_HTBX. The standard deviation from

Maximising K_{eff} only in Test Case 3

Algorithms	Best	Average	Standard Deviation	PPF
EDA_G	1.047680	1.047672	0.000008	1.187150
EDA_H	1.047680	1.047674	0.000007	1.187150
GA_HTBX	1.047270	1.046737	0.000378	1.189550

Minimising PPF only in Test Case 3

Algorithms	Best	Average	Standard Deviation	K_{eff}
EDA_G	1.149250	1.149255	0.000007	1.017840
EDA_H	1.149250	1.149260	0.000006	1.017840
GA_HTBX	1.149660	1.150320	0.000551	1.025590

Table 7.13: Maximisation of K_{eff} and minimising PPF only in Test Case 3 by EDAs and GAs. Results are from 30 independent runs, showing the best, average and standard deviation. The corresponding PPF when the best K_{eff} is found (w1 = 1), and the K_{eff} when the best PPF is found (w2 = -1), are listed in the last column.

30 independent runs also suggests that the tested EDAs are more robust and less disruptive search methods than the tested GAs. The averaged performances from 30 independent runs on both scenarios are plotted in figure 7.10.

It can be seen that the heuristic information used in EDA_H, the stand-alone PPF, did not help EDA_H outperform EDA_G. The averaged best solution found in 30 independent runs with random initialisation by EDA_H is slightly worse than for EDA_G. It is understood from previous test cases that EDA_H concentrates on an area indicated by the heuristic information, while EDA_G performs a better exploration. This local convergence feature in EDA_H, however, may not be desirable in some cases, such as minimisation of PPF, because two radically different LPs could have very similar PPF values.

From the two preliminary tests, both EDAs perform well in maximising K_{eff}, and EDA_H shows less variation between different independent runs. On minimising PPF, both EDAs managed to find the best solution. The heuristic information used in EDA_H did not improve the averaged performance.

By combining K_{eff} maximisation and PPF minimisation together using weights $w1 = 0.8$ and $w2 = -0.2$, a more complicated test problem is created. The EDAs are doing well in maximising K_{eff}, but in minimising PPF, the concern

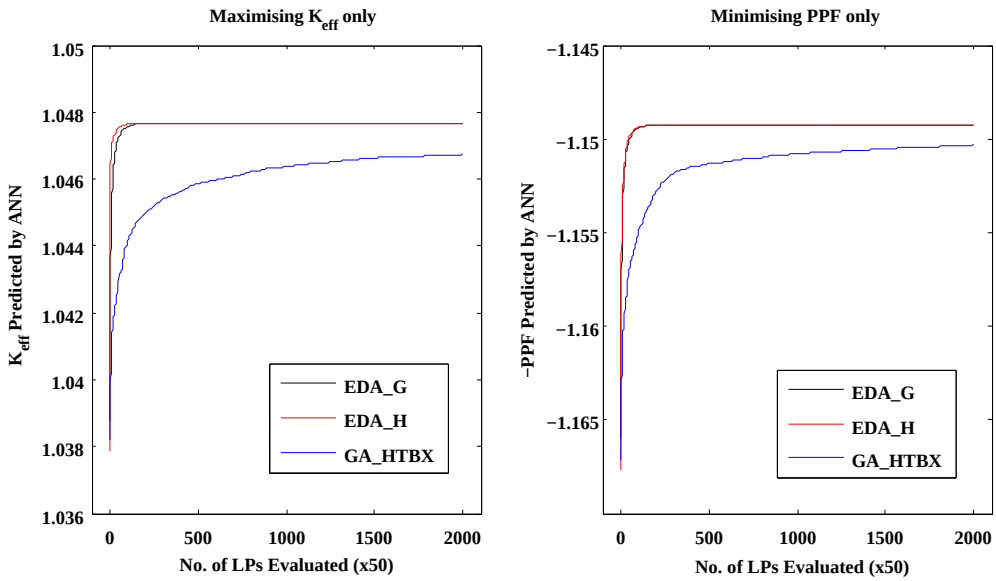


Figure 7.10: The averaged performance comparison between EDA_G, EDA_H and EDA_HTBX on Test Case 3, with only one objective at a time - Maximising K_{eff} to the left and minimising PPF on the right.

is that EDA_H's strong local search feature may restrict its exploration, and the best solutions found may consequently lack diversity.

The experimental results summarised in table 7.14 show a similar pattern compared to the results from Test Case 2. Both EDAs outperformed GA_HTBX in 100,000 function evaluations in terms of solution quality and convergence speed. This comparison shows that the tested EDAs are as good as, if not better than, the tested GA_HTBX in global search, given a reasonably large number of function evaluations. Figure 7.11 shows the averaged performance of 30 independent runs of EDAs and GA_HTBX and figure 7.12 compares the error bounds between EDA_H and GA_HTBX. It is clear that EDA_H converged much faster than the tested GA, and found better solutions.

In this case, EDA_G and EDA_H produce very similar results. The heuristic information, H, utilised in EDA_H did not have a great impact on the algorithm's performance, in terms of solution quality and convergence speed. This is expected, given the fact that EDA_H did not outperform EDA_G in minimising PPF alone.

Algorithms	Best	Average	Standard Deviation	K_{eff}	PPF
EDA_G	0.605366	0.605291	0.000080	1.046260	1.158210
EDA_H	0.605366	0.605293	0.000079	1.046260	1.158210
GA_HTBX	0.605058	0.604047	0.000578	1.046460	1.160550

Table 7.14: The best objective function value found by EDAs and GAs from 30 independent runs and their corresponding average and standard deviation. The corresponding K_{eff} and PPF are also shown.

The range of searched K_{eff}:		
Algorithms	Maximum	Minimum
EDA_G	1.046600	1.007610
EDA_H	1.046650	1.007780
GA_HTBX	1.046460	1.005450
The range of searched PPF:		
Algorithms	Maximum	Minimum
EDA_G	1.196800	1.152800
EDA_H	1.194560	1.152980
GA_HTBX	1.201160	1.152330

Table 7.15: The maximum and minimum K_{eff} and PPF values found by EDAs and GAs from 30 independent runs on Test Case 3 with the combined objective function F - $w1 = 0.8$ and $w2 = -0.2$.

There is a trade-off between the objective function and the constraint. A better local convergence, as in EDA_H, may not contribute significantly to exploring the trade-off.

From the K_{eff} and PPF values shown in table 7.15, it can be seen that GA_HTBX is exploring a considerably wider range of objective/constraint function space, while both EDAs focus on rather limited ranges, particulary EDA_H, which is restricted by the heuristic information. If this heuristic information does not improve the optimisation significantly, it might be switched off. For multi-objective problems in which the trade-off needs to be explored, EDA_G might be considered a better option than EDA_H, given the experimental results from Test Case 3. This will be discussed in section 7.6.2.

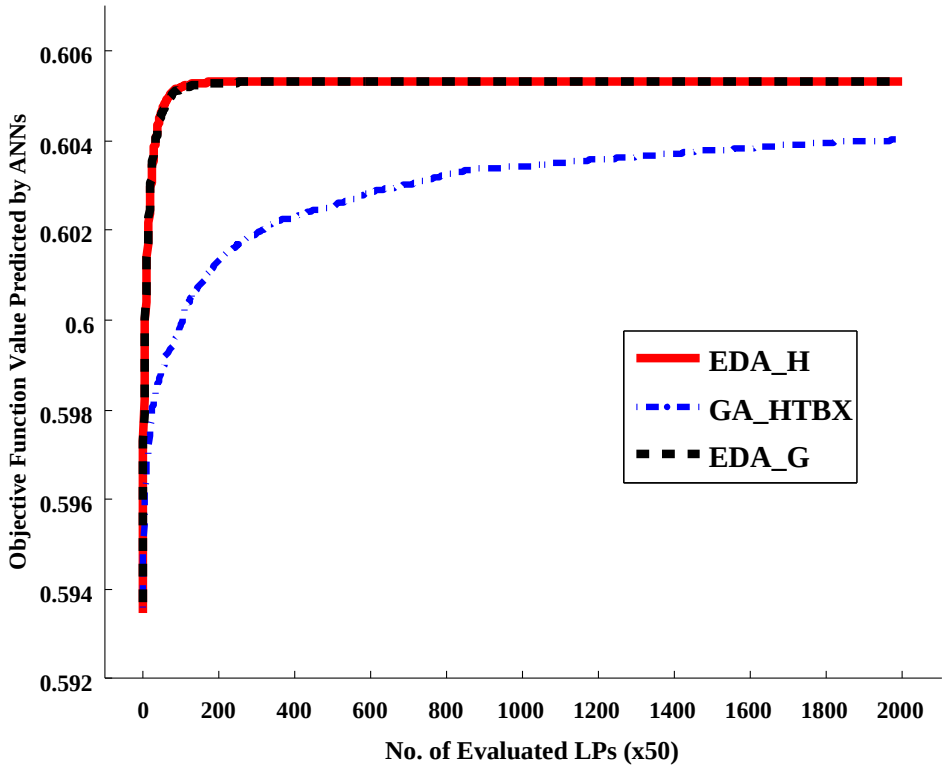


Figure 7.11: The comparison between EDA_H and GA_HTBX on the modified CONSORT case on the modified CONSORT case with power peaking constraint.

7.5 Parameter Sensitivity

It is known that the performance of evolutionary algorithms such as GAs can be sensitive to parameter settings. In this section, we present an empirical study of the parameter sensitivity of the EDAs when applied to LP optimisation problems.

We use Test Case 3 with the combined objective function F in equation 7.16 as the benchmark problem, because it is derived from a real-world reactor application and a typical power peaking constraint is included. The basic parameters settings are taken from table 7.9 and the population size, α, η, β and mutation rate are tuned separately. The experimental results are based on 20 independent runs. For each run, it starts with a random population generated on the fly.

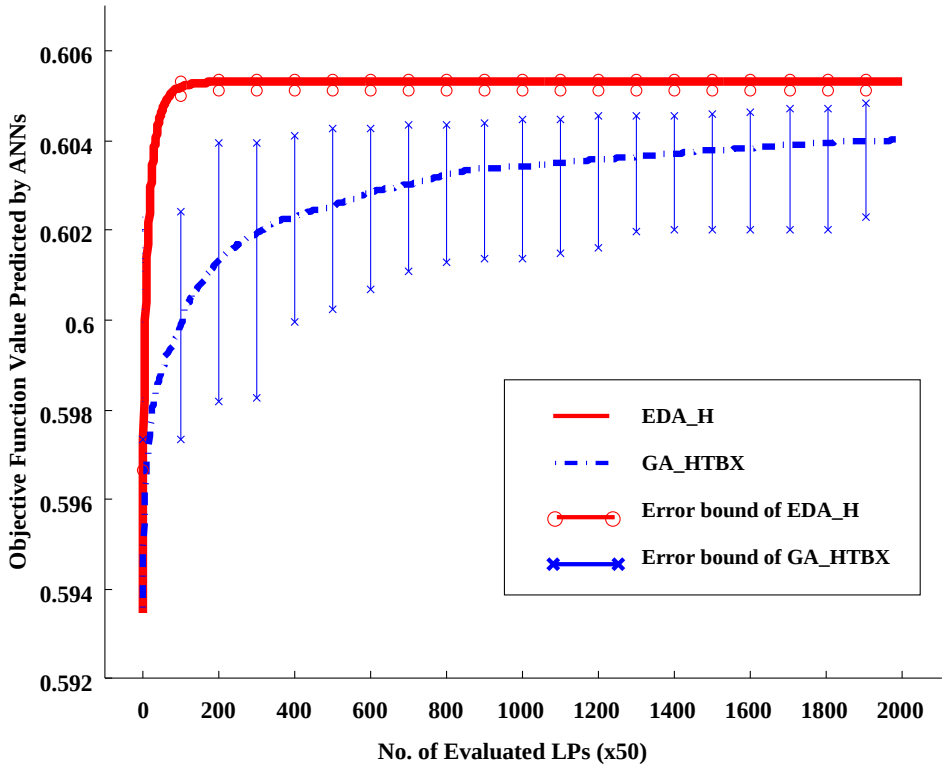


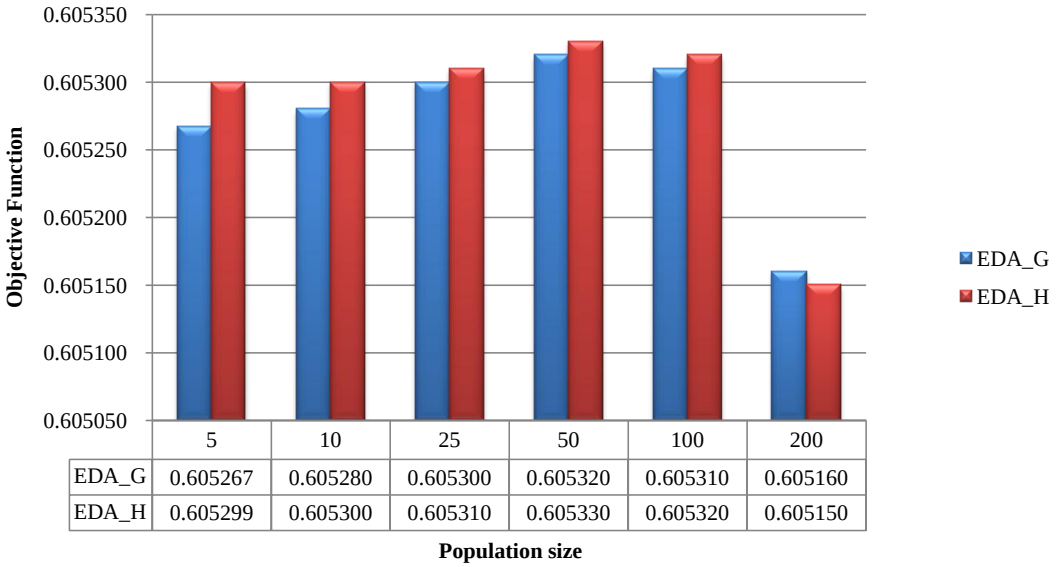
Figure 7.12: The averaged performance of EDAs and GAs against number of LP evaluations with error bounds on the modified CONSORT case with power peaking constraint.

7.5.1 Population Size

Population size is the number of candidates maintained in a population-based algorithm, e.g. a GA or EDA, in one time step or 'generation'. Generally speaking, a small population size may cause premature convergence; a large population size, on the other hand, will yield relatively slow convergence, but a better exploration is expected.

In Chapter 4 we mentioned that previous research showed that some EDAs converge to the global optimum on additively decomposable problems, given an infinite population size. In practice, we expect that the EDAs can generate reasonably good results within reasonable time and cost. The averaged best solution found for Test Case 3 using different population sizes in 20 independent runs are summarised in figure 7.13. Please refer to table 7.9 for the values of other control parameters.

A similar pattern can be seen from both algorithms, which is that as the



	5	10	25	50	100	200
EDA_G	0.605267	0.605280	0.605300	0.605320	0.605310	0.605160
EDA_H	0.605299	0.605300	0.605310	0.605330	0.605320	0.605150

Figure 7.13: The averaged best objective function value in 20 independent runs found by EDAs within 100,000 function evaluations but with different population sizes

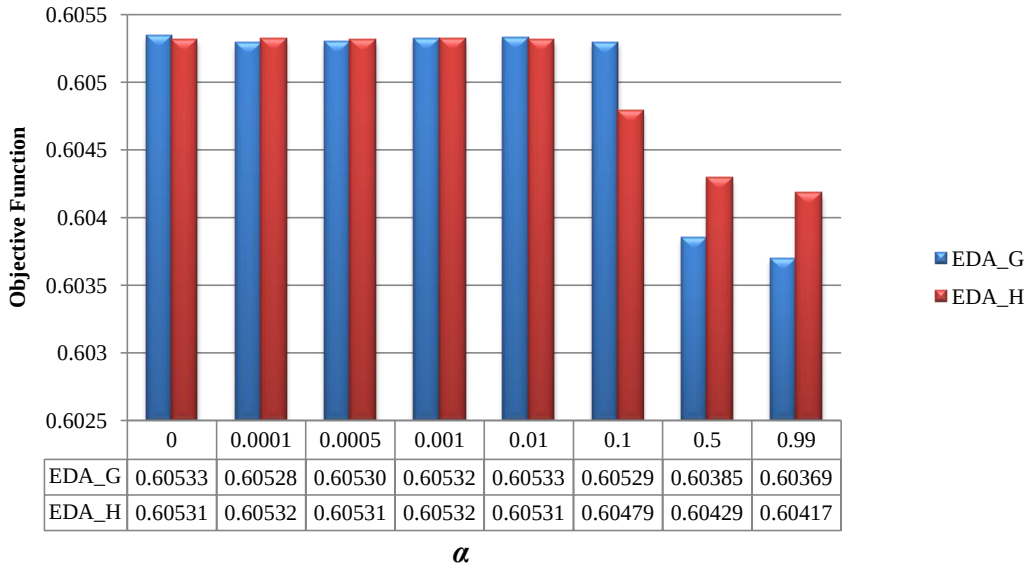
population size increases, the EDAs perform better until the population size becomes too large, i.e. over 100. A larger population size benefits the global search because the candidate pool contains more diversified individuals. However, to achieve a good result, more candidates will need to be tested, which can be very inefficient and computationally expensive. In this case, a population size between 25 and 100, e.g. 50, has worked well for both EDAs.

7.5.2 Incremental Learning Rate: α

The incremental learning rate α in the EDAs determines how the distribution model is being updated. Recall the following equation 7.20:

$$P_{t+1} = (1 - \alpha) \cdot P_t + \alpha \cdot X \tag{7.20}$$

where α controls how fast the distribution model P evolves with the statistical information in X. Normally the value of α is between 0 and 1. A small α can reduce the risk of early convergence and concomitantly slow down the exploitation.

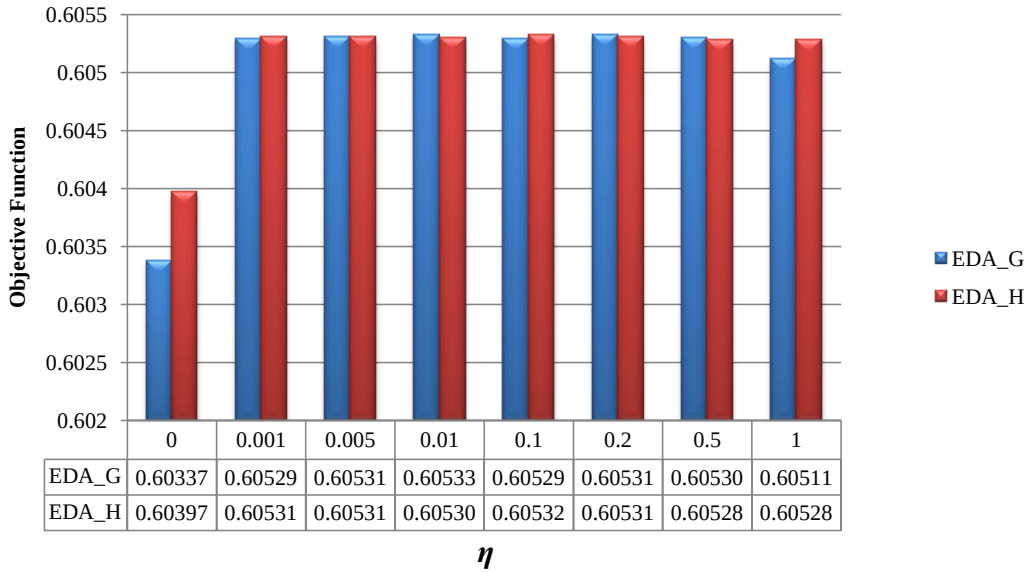


	0	0.0001	0.0005	0.001	0.01	0.1	0.5	0.99
EDA_G	0.60533	0.60528	0.60530	0.60532	0.60533	0.60529	0.60385	0.60369
EDA_H	0.60531	0.60532	0.60531	0.60532	0.60531	0.60479	0.60429	0.60417

Figure 7.14: The averaged best objective function value in 20 independent runs found by EDAs within 100,000 function evaluations but with different values of α

It should be noted that the value of α is related to the information in X. In PBIL, X is just the current best solution. In UMDA, X contains the information from the top N best solutions or N selected solutions. When N is equal to the population size, as it is in all of our test cases, the statistical information in X should be very similar to P_t. This will slow down the convergence and therefore reduce the risk of local convergence.

In Zhang (2004b), the author concluded that UMDA and similar types of EDAs may be trapped at every local optimum. Since the developed EDAs in this work use a similar type of probability model, theoretically a small value for α should be used, so that the distribution model is updated slowly and it will help the algorithm to explore the search space better. The test results in figure 7.14 show that a small α did perform well for both EDAs. The algorithms perform well even when α is 0, due to our use of the elitism strategy and heuristic information (in the case of EDA_H) to improve the optimisation. Please refer to table 7.9 for the values of other control parameters.



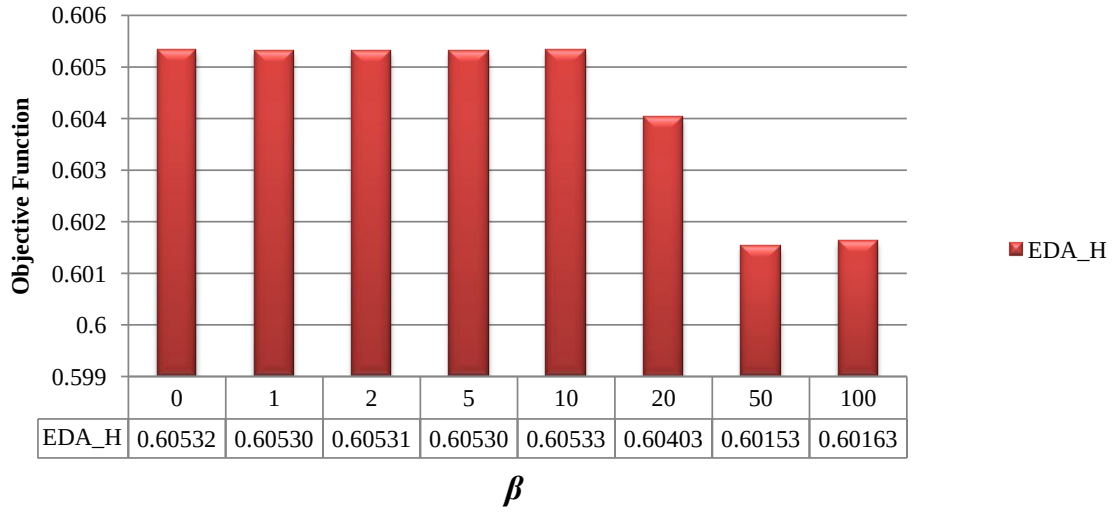
	0	0.001	0.005	0.01	0.1	0.2	0.5	1
EDA_G	0.60337	0.60529	0.60531	0.60533	0.60529	0.60531	0.60530	0.60511
EDA_H	0.60397	0.60531	0.60531	0.60530	0.60532	0.60531	0.60528	0.60528

Figure 7.15: The averaged best objective function value in 20 independent runs found by EDAs within 100,000 function evaluations but with different values of η.

7.5.3 Elitism Rate: η

In evolutionary algorithms, the current best solution can be kept in the population all the time. This is known as the elitism strategy. In our EDAs, this is converted into an elitism rate η. By using η, the information contained in the current best is applied to the distribution model P directly and weighted by η. This parameter helps local convergence when the current best is near the centroid of the current population. If the current best is far away from the population, η will drive the whole population in that direction. Figure 7.15 shows the experimental results of using different values of η in Test Case 3. Please refer to table 7.9 for the values of other control parameters.

A small η would not have sufficient impact, but a larger η may worsen the performance, as it reduces the candidates' diversity and causes premature convergence. This pattern is clearly shown in EDA_G's results, and 0.01 appears to be a well-balanced value. For EDA_H, it is less sensitive to a small η, because even if η is not working, the heuristic information will still be able to accelerate the convergence.



	0	1	2	5	10	20	50	100
EDA_H	0.60532	0.60530	0.60531	0.60530	0.60533	0.60403	0.60153	0.60163

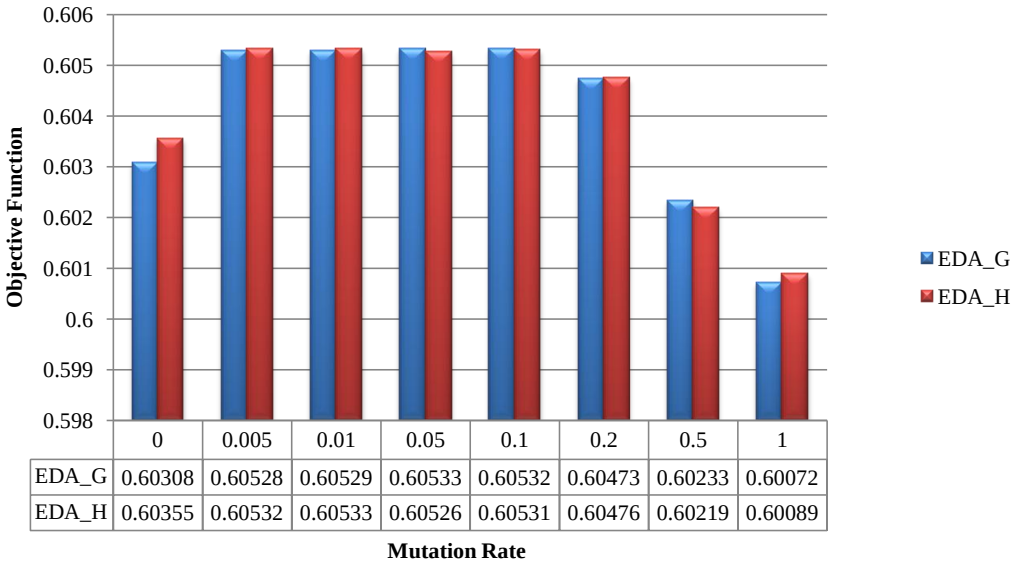
Figure 7.16: The averaged best objective function value in 20 independent runs found by EDAs within 100,000 function evaluations but with different values of β.

7.5.4 Heuristic Information: β

The heuristic information used in this work and the way it is used have considerable effect on the EDAs performance. This information improved the convergence speed as well as the robustness of the EDAs. The key to its success is the use of the contributions of the building blocks to the objective function. However, this information may cause early convergence and therefore may not be able to explore the trade-off well when applied to a multi-objective problem.

It should be noted that from our test cases, some values in the H matrix are small and the difference between different entries is not significant enough, which can result in using a large β. We suggest ranking all entries, e.g. a 24×35 matrix will be filled with integers between 1 to $24 \times 35 = 840$. By doing this, the difference between any fuel element-channel assignment is considerably larger and therefore a much smaller β can be used, e.g. in Test Case 3, $\beta = 2$. From our experiments, integer values between 1 to 20 are sufficient.

As shown in figure 7.16, EDA_H is less sensitive to smaller β but if β is too



	0	0.005	0.01	0.05	0.1	0.2	0.5	1
EDA_G	0.60308	0.60528	0.60529	0.60533	0.60532	0.60473	0.60233	0.60072
EDA_H	0.60355	0.60532	0.60533	0.60526	0.60531	0.60476	0.60219	0.60089

Mutation Rate

Figure 7.17: The averaged best objective function value in 20 independent runs found by EDAs within 100,000 function evaluations but with different mutation rates.

large, e.g. $\beta = 20$, the performance drops significantly, because it can easily lead to a premature convergence. In this case, the heuristic information did not improve the performance against EDA_G, so even if $\beta = 0$, the algorithm still performs well as it is essentially EDA_G. The values of other parameters are shown in table 7.9.

7.5.5 Mutation Rate

The mutation used in the EDAs is a swap mutation applied to a candidate LP. The purpose of mutation is to explore the whole search space in a random manner, and to prevent the premature convergence problem. From previous research in evolutionary algorithms, performing a random mutation with a low frequency normally helps the algorithms to explore the entire search space. If the mutation rate is too low, it may not explore as expected; if it is too large, the algorithm becomes too disruptive and does not converge well. Figure 7.17 shows that a mutation rate between 0.005 and 0.01 is a well-balanced value. Please refer to table 7.9 for the values of other control parameters.

7.6 Constraints and Multiple Objectives

In many real-world applications, there are often constraints and/or multiple objectives involved in problem solving. An example is the PPF constraint in fuel loading optimisation. In this section, we will briefly discuss how to apply the EDAs to solve constrained or multi-objective optimisation problems, with some well studied technologies. The purpose of this section is exclusively to demonstrate how they can be utilised in the EDAs. Extensive research on the EDAs for multi-objective optimisation is not included in the current work but will be conducted in the near future.

7.6.1 Handling Constraints

When a constraint is not satisfied during optimisation, the corresponding solution can be discarded. Sometimes this may not be desirable as more solutions may be needed for exploration. Two commonly used methods for constraint satisfaction in an evolutionary optimisation algorithm are repairing and the use of penalty functions.

Repairing is the manual adjustment of the proposed solutions in order to ensure that they satisfy the constraint(s). Specially designed repairing operators may require extra development work and computational cost, and they do not necessarily improve the optimisation. Using a penalty function consists of adding a negative effect to the objective function given the constraints, hence making the proposed solution less favourable. Penalty functions combine the constraints and the objective function and effectively form another optimisation problem. It is relatively easy to implement, but again it does not always produce better results for the original problem.

Choosing a constraint handling method is very problem-specific. In this work, we used both methods. To ensure that fuel channels 6 and 15 are always loaded with two specified fuel elements, a repairing operator is hard-coded in the algorithms to check and repair every LP generated (especially after mutation). In Test Case 3, we

combine two weighted terms into one single objective function and the PPF term can be regarded as a penalty function.

The constraint can be strengthened or relaxed by adjusting the weights for the main objective and the penalty term. The combined objective function will still be optimised, but the contributions from the two terms will vary. In some cases, a constraint may be considered as another objective and hence the algorithm is effectively solving a multi-objective problem. For example, in Test Case 3, the PPF can be treated as another objective to be minimised.

7.6.2 Handling Multi-objective Problems

For multi-objective optimisation problems, the decision maker normally requires a set of Pareto optimal solutions (Miettinen (1999)), which ideally approximates the trade-off or the Pareto front of the problem, in order to choose the most practical solutions. There are various strategies for multi-objective optimisation, including decomposition and non-dominated search.

Decomposition

Decomposition is a conventional and straightforward approach for multi-objective optimisation (Zhang and Li (2007)). It consists of decomposing the multi-objective problem into a number of scalar optimisation problems and then solve them simultaneously or separately. As pointed out in Zhang and Li (2007), a Pareto front could be approximated by the solutions for a number of sub-problems decomposed from the original multi-objective problem. The authors concluded that a decomposition based evolutionary algorithm could outperform some well established genetic algorithms using non-dominated sorting search, with less computational complexity.

Another motivation for the decomposition methods is that the Pareto front sometimes may consist of a very large, or even infinite number of solutions. It is very difficult to find the whole Pareto optimal set. In practice, it is reasonable only to have a set of solutions distributed evenly on the Pareto front. If a proper

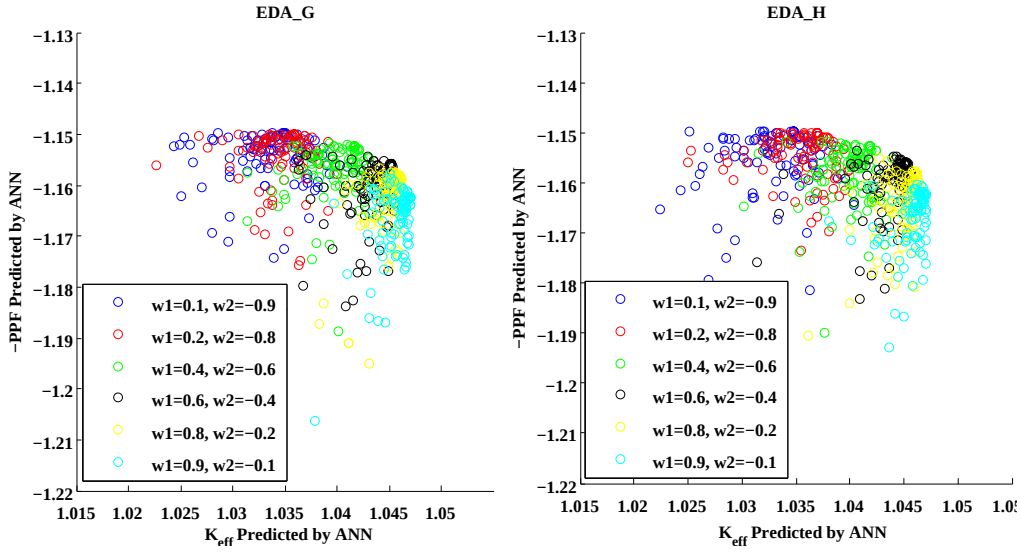


Figure 7.18: Using a set of weight vectors to explore the trade-off in the EDAs applied to Test Case 3. For each set of weights, 100 solutions from the last two generations are plotted.

decomposition method is used, it is possible to achieve a reasonably-sized set of solutions that are more evenly distributed on the Pareto front.

A few decomposition and aggregation methods are introduced and applied in Li and Zhang (2009), and among them, the weighted sum method is the most straightforward and easy to implement. For example, in Test Case 3, different pairs of $w1$ and $w2$ can be used to approximate the whole multi-objective problem.

In Zhang and Li (2007) and Li and Zhang (2009), the advanced decomposition method is used and the sub-problems are optimised simultaneously with communications between neighbour sub-problems. However, in this section we will only demonstrate how to use EDAs with simple decomposition to find the approximate Pareto front. The development of more advanced EDAs will be part of future work.

To generate a set of solutions approximating the Pareto front based on weighted sum decomposition, a set of weight vectors are required to explore the trade-off. In figure 7.18, EDA_G and EDA_H are applied to Test Case 3 again with the same parameter settings as in table 7.9 in the previous section, but with 6 different sets

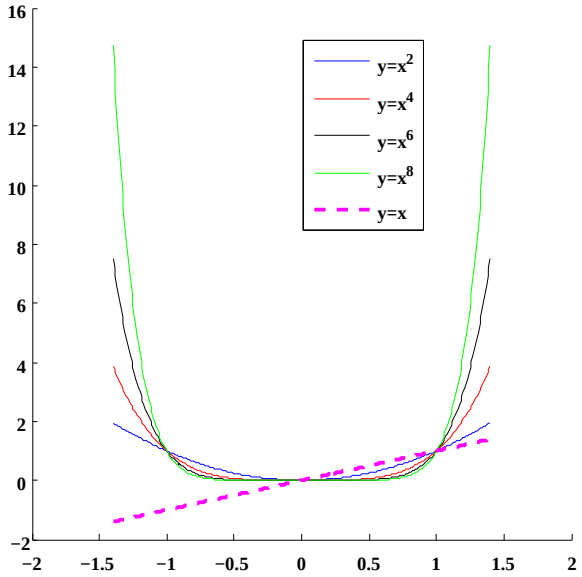


Figure 7.19: Illustration of linear constraint/objective rescaling and using a 'bucket' function

of $w1$ and $w2$ values. For each set of weights, the last two generations of the population (100 individuals) are plotted. It can be observed that as the weights change gradually, the final solutions are pushed along a reasonable trade-off between K_{eff} and PPF.

EDA_G and EDA_H show a similar trade-off approximation in this case. The solution distribution of EDA_G is slightly more even than for EDA_H, and the trade-off formed is also slightly smoother. This is caused by the strong local search feature in EDA_H. Having said that, the differences are not at all significant in this case.

In practice, it is also common to adjust the constraint term in a non-linear manner. For example:

$$F = w1 \cdot K_{eff} + w2 \cdot PPF^n \tag{7.21}$$

where n is an exponential scalar for adjusting the impact of PPF. Compared to the weighted sum method, this method offers a much stronger scaling on the PPF term. From figure 7.19, the 'bucket' shaped functions are able to scale the magnitude

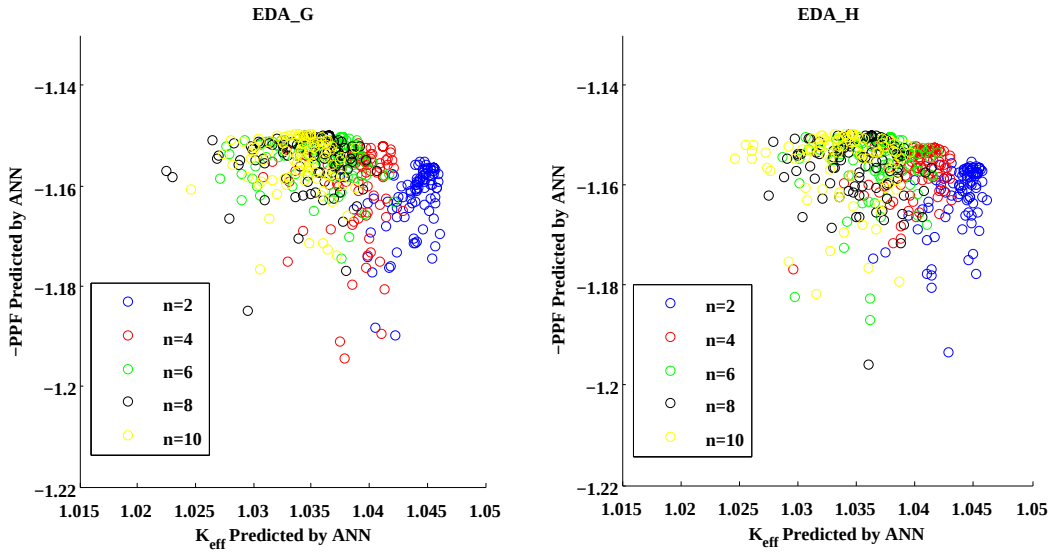


Figure 7.20: Handling the PPF - the second objective or a constraint using an exponential term in EDAs for Test Case 3 as shown in equation 7.21. For each n, 100 solutions from the last two generations are plotted. The weights are always set as: $w1$=0.8, $w2$=-0.2

radically, and therefore provide an alternative control for finding trade-offs for multi-objective optimisation.

EDA_G and EDA_H are tested using different n as shown in figure 7.20. The weights are always set as: $w1 = 0.8$ and $w2 = -0.2$. Other parameters are the same as in table 7.9. The last two generations of populations are plotted. Although the difference between figure 7.18 and 7.20 is not very significant, it is obvious that for extensive search for Pareto fronts in real-world applications, rescaling parameters using weight vectors and 'bucket' functions are both effective methods.

One of the disadvantages of running multiple optimisation routines with different weight vectors as shown above is probably the large number of objective function evaluations. However, as we saw from previous experiments, EDA_H is able to converge very fast even with a small population size. It would be interesting to see if repeating EDA_H with a limited number of objective function evaluations and different weights and exponential scaling can approximate the Pareto front.

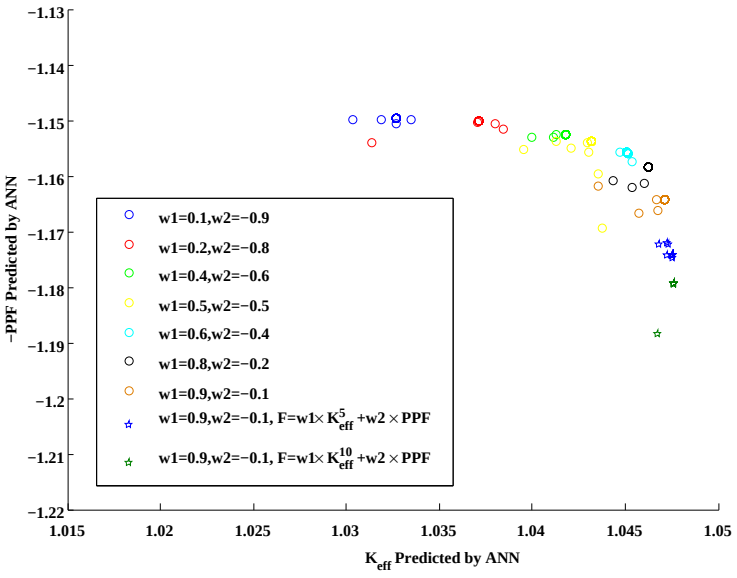


Figure 7.21: Using EDA_H with a small population size (10) to approximate the Pareto front.

In figure 7.21, EDA_H is tested with a population size of 10 with all other parameters the same as in table 7.9. The weights are displayed in figure 7.21. The last two sets of data are produced with an exponential factor of 5 and 10 applied to the K_{eff} term, respectively. The final two generations of populations for each run are plotted. It is shown that each run produces a set of solutions that are much less diversified, but due to the decomposition, it still approximates a reasonable Pareto front.

From the discussions above, without any modification, the EDAs for scalar optimisation can be applied to multi-objective problems by decomposing the problem and simply re-running the algorithm. Weighted sum and exponential rescaling can be used to adjust the algorithm for approximating the Pareto front. If the computational cost is limited, the EDA_H method can be used with a small population size without compromising the performance too much.

Using Non-dominated Sorting in the EDA

As discussed in the previous section, decomposition is a conventional method for multi-objective optimisation, but it is also a rather recent and active research topic for its application in multi-objective evolutionary algorithms. Most of the well established multi-objective evolutionary algorithms are focusing on solving the multi-objective problem as a whole. It is also in the researchers' interest to develop a method that generates more non-dominated solutions. For these reasons, some algorithms based on domination were developed; the Non-dominated Sorting Genetic Algorithms II (NSGA II) in Deb et al. (2002) is one of the best-known.

In this section, the EDA algorithm is combined with a well established non-dominated sorting operator and tested, in order to demonstrate its capability for using a domination based method.

The key concerns of NSGA II and other domination based searching approaches are the methods of sorting, selecting and using the non-dominated solutions. Since the EDAs have a very similar algorithm structure to the GAs, a non-dominated sorting-selection operator can be used to replace the existing selection method with very little effort.

We implemented the non-dominated sorting operator described in Deb et al. (2002), and incorporated it in EDA_G. EDA_H is not used because it is specially designed for scalar optimisation and not good at maintaining solution diversity unless further development is performed.

The modified EDA_G algorithm still uses the same probability model:

$$P_{(t+1)} = (1-\alpha) \cdot P_{(t)} + \alpha \cdot X + \eta \cdot X_{non-dominated} \tag{7.22}$$

where X is the selected LPs based on their Pareto ranking (not a weighted sum of two objectives) using a proportional selection method as in the original EDA_G algorithm; $X_{non-dominated}$ represents the information extracted from the external population that stores all the known non-dominated LPs. The rest of the algorithm

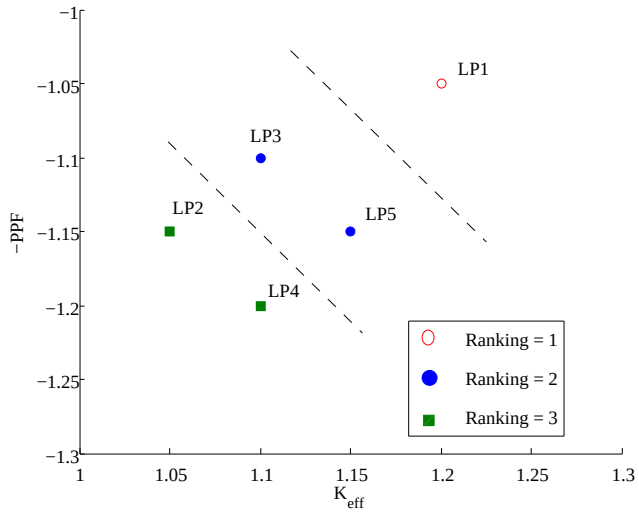


Figure 7.22: A set of example LPs plotted to illustrate their Pareto ranking.

remain unchanged. The ranking method and modifications to the elitism term ($X_{non-dominated}$) are explained below with a simple example.

Given a population of candidate LPs in this case, we compare both the objectives, the K_{eff} and the PPF, of each candidate LP with the corresponding objectives of all other LPs. If neither its K_{eff} nor the PPF are both worse than any other candidate (otherwise known as a non-dominated LP), this LP is then given a ranking number '1'. After comparing all the LPs with each other, the ones with ranking number '1' form the current Pareto set.

The first Pareto set is then excluded from the current population and this process is repeated until no LP is left in the candidate pool. The whole population is divided into a number of sub-sets, and each of them is associated with a unique ranking number. For example, the Pareto ranking of a set of example LPs is shown in table 7.16. These LPs are also plotted in figure 7.22 to illustrate the ranking.

In this example, it is clear that LP1 dominates the rest of the LPs because it has the best K_{eff} and PPF. LP1 is then given a ranking number '1' and excluded from the candidate pool. Among the rest of the LPs, because LP3 has a better

	K_{eff}	PPF	Pareto Ranking
LP1	1.20	1.05	1
LP2	1.05	1.15	3
LP3	1.10	1.10	2
LP4	1.10	1.20	3
LP5	1.15	1.15	2

Table 7.16: The Pareto ranking of a set of example LPs.

K_{eff} than LP2 and the same PPF value as LP2, it dominates LP2. Similarly LP3 also dominates LP4. LP5 is not dominated by LP3 due to a better K_{eff} value, and it dominates LP2 and LP4 as LP3 does. Both LP3 and LP5 are then given a ranking number '2' and excluded from the candidate pool. Since neither LP2 nor LP4 dominate each other, they are both given a ranking number '3' and the sorting is completed.

After the sorting, every LP is assigned a ranking number. We use their ranking numbers as their new fitness values. These fitness values are then used in a standard proportional selection operator. The higher the ranking (smaller ranking number), the more favourable a candidate is. The selected LPs are then used in the way described in Chapter 4, section 4.2, to form the X in equation 7.22.

To calculate the $X_{non-dominated}$ in equation 7.22, another commonly used technique for multi-objective optimisation is also implemented, which is to maintain an external population that stores all the non-dominated solutions found during the search. In each generation, the LPs with ranking number '1' are added to the external population, if and only if they are not dominated by any LP that is already in the current external population.

Since there can be more than one non-dominated LP in the external population, a simple arithmetical averaging method is applied. For example, given an external population with three LPs, encoded in a binary vector form:

$$
\begin{aligned}
{}^{1}x_{non-dominated} &= [1, 0, 1] \\
{}^{2}x_{non-dominated} &= [1, 1, 1] \\
{}^{3}x_{non-dominated} &= [1, 0, 1]
\end{aligned}
\tag{7.23}
$$

The $X_{non-dominated}$ is defined as:

$$
X_{non-dominated} = \frac{{}^{1}x_{non-dominated} +^{2} x_{non-dominated} +^{3} x_{non-dominated}}{N} \tag{7.24}
$$

where N is the number of LPs in the external population, and in this case N=3. Hence:

$$
X_{non-dominated} = [\frac{1+1+1}{3}, \frac{0+1+0}{3}, \frac{1+1+1}{3}] = [1, \frac{1}{3}, 1] \approx [1, 0.33, 1] \tag{7.25}
$$

Then this $X_{non-dominated}$ is ready to be used in equation 7.22. The modified algorithm is applied to Test Case 3 with the parameters settings listed in table 7.9.

Experimental results for this modified algorithm are shown in figure 7.23 and figure 7.24. In figure 7.23, the last two generations of populations (the last 100 solutions) with and without the use of the η term are compared. It is clear that the population is very diverse without using η, but using η results in finding some better solutions at the cost of less exploration along the trade-off. In figure 7.24, the external populations with and without the use of the η term are compared. It shows a similar pattern to that in figure 7.23.

In this section, we demonstrated that the EDAs can also be used for multi-objective optimisation problems by using the established non-dominated sorting selection method.

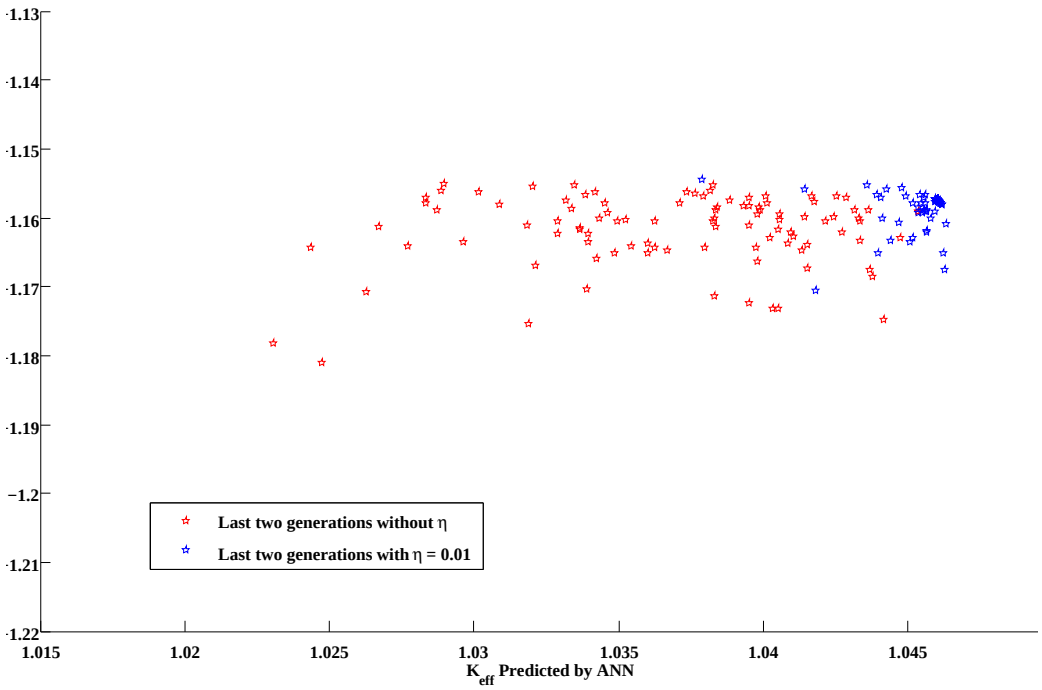


Figure 7.23: The modified EDA with non-dominated sorting selection operator for Test Case 3. The algorithm has been applied with and without the η term, and the last two generations of populations are plotted.

7.7 Summary

In this chapter we developed and applied the EDAs to the reactor fuel management optimisation problems. The test results are compared to the current state-of-the-art GA_HTBX. The EDA_G and EDA_H algorithms achieved better results in terms of solution quality, convergence speed and stability. However, the solutions diversity in the EDAs is not as good as in the GAs. To understand the algorithms better, the parameter sensitivities were also studied based on extensive experiments. Finally, we discussed how to handle constraints and multiple objectives in the EDAs.

It was observed that EDA_G and EDA_H produce faster convergence than the benchmark GA_HTBX. The original reason for using the EDAs was to improve the local convergence of the black-box-style optimisation algorithms without sacrificing their global search capacity. From the experimental results, the EDAs showed a better local search ability compared to the benchmark GA. Although both the EDAs found slightly better solutions than the GA_HTBX after 100,000 function evalua-

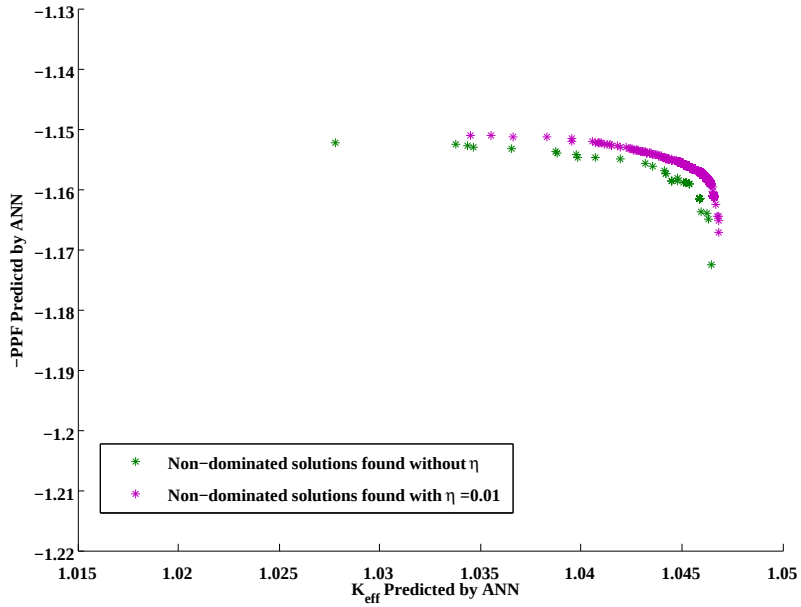


Figure 7.24: The modified EDA with non-dominated sorting selection operator for Test Case 3. The algorithm has been applied with and without the η term. The external population of all non-dominated solutions found are plotted.

tions, the GA_HTBX searches a considerably wider objective/constraint space.

For multi-objective optimisation, both decomposition and non-dominated sorting strategies can be used in the EDAs with little effort. It is shown that using multiple instances of EDA_H with limited population size could find the approximate Pareto front with acceptable computational cost. Meanwhile, by using the non-dominated sorting selection operator, the modified EDA can generate a set of non-dominated solutions in a single run.

The promising results encourage further research on the application and the development of the EDAs for both scalar and multi-objective optimisation problems. It should be noted that multi-objective optimisation is a difficult research topic in its own right. In this work, we only demonstrate how to use the EDAs to solve this kind of problem with some well studied methods, without extensive performance analysis, improvement or comparison.

Chapter 8

Conclusions and Future Work

8.1 Summaries

Chapter 1: Introduction

We presented an outline of the thesis and overviews of all chapters.

Chapter 2: Nuclear Reactor Fuel Management Optimisation

The reactor fuel management optimisation problem is reviewed with the industry background, focusing on the in-core fuel loading-pattern optimisation problem.

The objective functions commonly used are introduced, including the cost efficiency, the average energy extraction per fuel element, the core power distribution, and neutron emission rate in certain fuel channels (for research reactors).

The search space, i.e. the number of possible loading patterns, is estimated from the size of the reactor and the fuel inventory. Considering typical power/research reactors, the search space of LP optimisation can vary between 10^{10} and 10^{100} solutions.

The constraints in fuel management optimisation are discussed, including hard operational constraints and soft constraints regarding safety issues.

Chapter 3: Review of Previous Work

A review of previous work done for reactor loading-pattern optimisation is pre-

sented in this chapter. The algorithms are classified as 'white box' and 'black box' approaches.

The 'white box' approach requires good knowledge of the reactor's neutronics model, which can be reactor-dependent and/or simulation-model-dependent. A considerable amount of work has been established by explicitly mapping a predefined objective function with respect to in-core reactivity (K_∞) distribution/profile, normally in mathematical equations, then using conventional optimisation methods such as steepest descent, linear programming and Gauss methods. These methods are stable but not readily generalisable. The resulting loading pattern can be in numerical form, which needs to be rounded to integer form in some cases. This introduces additional accuracy loss.

The 'black box' methods, including SA and GAs, are briefly introduced. They can be easily implemented without extensive problem-specific knowledge, and are readily generalisable. However, a large number of objective function evaluations may be required, which could be very computationally expensive.

The current state-of-the-art Genetic Algorithms (GAs) with Heuristic Tie Breaking Crossover (HTBX) are introduced in more detail, because they will be used as the benchmark algorithm solving the LP optimisation problems.

Chapter 4: A Review of the Estimation of Distribution Algorithms

The Estimation of Distribution Algorithms (EDAs) is reviewed as a new development in evolutionary computation. We briefly introduce the background of the emergence of the EDAs, followed by an introduction of a basic EDA algorithm as it solves a simple optimisation problem.

Variants of the EDAs are introduced according to the probability distribution model used, which is the key concern in the EDAs. Probability models without variable dependence are easier to understand and implement, but the variable dependence may be better at indicating the promising areas in the search space. Using variable dependence in an EDA's probability model is more computationally expen-

sive than the ones without dependence.

How to apply the basic EDAs to real-world discrete and continuous problems is introduced, followed by discussions regarding the current theoretical work on their convergence and complexity.

Chapter 5: Estimation of Distribution Algorithms for the Travelling Salesman Problem

A few EDAs, with and without performance enhancements, are proposed for the classical combinatorial optimisation problem, the Travelling Salesman Problems (TSPs). Numerical experiments are done by applying the EDAs to a set of analytical TSPs and benchmark GAs with different crossover operators (Poon (1992) and Poon and Carter (1995)).

The EDAs showed good performance against the tested GAs. It is observed that the EDAs with an elitism strategy and using heuristic information outperform other test algorithms in terms of solution quality and standard deviation.

Chapter 6: Estimation of Distribution Algorithms for Fuel Management Optimisation

We demonstrate how to apply the EDAs to the reactor fuel loading-pattern optimisation problems by going through a simple example without a specified objective function, but focus on explaining the workflow of the algorithm and the use of the probability model, possibly with heuristic information.

Chapter 7: Applications and Results

The proposed EDAs were applied to three test cases derived from the CONSORT reactor. Having briefly introduced the CONSORT reactor, three test cases were set up on different core states. Experiments were carried out and the results were compared with the chosen benchmark GAs.

The first test case was based on a clean-core state without control rods and

metal clad, only containing fuel and water (coolant and moderator), so the problem is expected to be more regular. A larger hypothetical fuel inventory, compared to the original one, is used, and the fuel elements are categorised into five fuel types, according to their designs.

The second test case was set up on a more complicated core state with the presence of control rods and metal clad. The variations between fuel elements of the same fuel types were also considered. Some modifications were made to the problem-encoding and algorithm.

The third test case is very similar to Test Case 2 except the modified fuel inventory and the objective function, which was combined with a power peaking constraint. The control parameters' sensitivity is studied explicitly by performing extensive experiments. In addition, some commonly used techniques for multi-objective optimisation are introduced and applied to the proposed EDAs.

Chapter 8: Conclusions and Future Work

The thesis is summarised based on the research and experiments done. Conclusions are drawn and possible future work is suggested.

8.2 Conclusions

In this thesis we described the application of the EDAs to reactor fuel management problems. Numerical experiments were carried out to investigate EDAs' performance on reactor loading-pattern optimisation. We compared the results from EDAs against those obtained from the benchmark GAs that employed well established crossover operators (Poon (1992) and Poon and Carter (1995)). Numerical experiments were performed based on trained ANNs instead of the accurate simulation code EVENT. Using an ANN reduces CPU times significantly for objective function evaluations, thereby allowing extensive numerical tests. It has been shown that the EDAs provide efficient, accurate and robust solutions for typical reactor loading pattern optimisation problems. In addition, the EDAs can be readily extended to

multi-objective optimisation problems with reasonable effort by implementing some commonly used techniques (decomposition and non-dominated sorting). However, their performance on multi-objective reactor loading pattern optimisation is yet to be verified with further study, development, and comparison in the near future.

The EDA approach has been shown to have several advantages compared to other existing methods applied to the same class of problems. Firstly, a basic EDA is a 'black box' method that does not require problem-dependent information. Unlike 'white box' approaches, EDAs are widely applicable to any similar problems or other combinatorial optimisation problems with little effort.

Secondly, the results from the test cases presented here showed that both EDAs with or without problem-dependent information (e.g. the stand-alone K_{eff} with fuel coupling) are very competitive algorithms compared to the benchmark GAs applied to the same problem. With the help of the stand-alone K_{eff} with fuel coupling as heuristic information, EDA_H outperformed all the other algorithms. The variance among different independent EDA_H runs is also much smaller than for tested GAs.

We demonstrated how the novel application of stand-alone K_{eff} with fuel coupling in the EDA_H algorithm improves the LP optimisation. This information includes physical properties of fuel assemblies, the effect of fuel channel positions and the effect of neighbourhood fuel assemblies, and can therefore greatly improve the performance of the EDAs. However, using heuristic information will restrict the algorithm from exploring a wider range of search space, which may not be ideal when a constrained optimisation task requires a good exploration over the tradeoff.

For constraints and multiple objectives, decomposition and non-dominated sorting based methods are shown to be easily compatible within the proposed EDAs framework. The preliminary results and current research work in this field (e.g. Zhang et al. (2008)) encourage further development of more advanced EDAs for multi-objective optimisation problems.

We recommend the application of the EDAs to more complicated test cases and real-world problems. We believe that further development of more advanced

EDAs is very promising and will make a significant contribution to the engineering community as well as to evolutionary algorithm researchers.

8.3 Future Work

Having successfully applied the EDAs to the test cases derived from the CONSORT reactor, we propose the following future work.

8.3.1 EDAs for Multi-Objective Problems

The majority of this work focuses on scalar optimisation problems because the optimisation task for the test reactor does not involve multiple objectives. Although we did some preliminary tests on multi-objective optimisation in Chapter 7, it is very important to perform a more comprehensive study on multi-objective EDAs.

This will involve testing/benchmarking the proposed EDAs on well understood test cases, and from the results and our experience on scalar optimisation, developing new EDA based algorithms.

The multi-objective test problems will be collected from the literature. It might be difficult to find real-world applications but there are plenty of analytical problems. For example, in Li and Zhang (2009), a set of continuous test instances with prescribed and complicated Pareto sets (Pareto optimal decision variables) is established. Extensive testing can be performed on these problems to understand the EDAs better and possibly inspire new ideas.

From our test results, it is clear that the EDAs have better local convergence characteristics than conventional GAs. This is desirable when reasonable solutions are required given limited resources. However, it precludes a better exploration. This disadvantage might be even more significant when applied to multi-objective optimisation problems. This will need to be addressed by modifying the algorithms.

We used a very simple decomposition strategy in the EDAs to search for Pareto optimal solutions, which is to simply re-run the algorithm with different weights or

exponential scaling for each objective. However, advanced decomposition methods and algorithms can be developed for this purpose. In Zhang and Li (2007), an evolutionary algorithm based on decomposition is developed for multi-objective optimisation problems. A set of pre-defined sub-problems are solved simultaneously and the current best solution of a certain sub-problem is passed to neighbour sub-problems and is potentially a good solution to them too.

Similar ideas can be used in the EDAs. A possible way is to start multiple instances of EDA_H with different weight vectors and consider each sub-population as an individual 'island' but with communications between them, passing the current best on to the neighbouring 'island'. Since EDA_H could perform well with small population size, the computational cost can still be acceptable. The other advantage is that this 'island' model can be easily be extended to parallel computing techniques.

We understand that the proposed EDAs are driven by the current best and/or the pre-defined heuristic information. Therefore the model sampling operator is likely to generate many solutions in a relatively small area in decision variable space. Although decomposition (e.g. different weights) can be used for this issue, it is still desirable to have a built-in method for maintaining population diversity. This can be done by introducing a distance measurement between the candidate solutions, e.g. the crowding distance introduced in Deb et al. (2002).

Another approach might be the use of an advanced distribution model in the EDAs, such as multiple distribution models in one algorithm. These distribution models can be pre-defined or learnt online during the optimisation process. The latter might need a classification routine to classify the available solutions and then build/update a corresponding sub-model.

Also, the development and application of the EDAs with reasonable variable dependencies that are able to model/sample from a structured distribution of promising solutions may be worthy of further investigation.

8.3.2 Surrogate Modelling in the EDAs

In this work, the use of ANNs is very important as they estimate K_{eff} and PPF given an LP within a fraction of a second, which makes extensive experiments possible. The ANNs can be regarded as a surrogate model for the tested reactor, and it is fair to say that they can approximate objective search space if sufficient training samples are available.

Investigating the EDAs performance using a surrogate model or on a response surface could save a considerable amount of time, particularly in the algorithm development/testing period. However, the surrogate objective function model using neural networks may affect the search due to the modelling bias and variance. It may be worth finding out how the model bias can affect the optimisation in order to avoid search bias. In practice, the surrogate model should be used together with an accurate model to avoid search bias.

Alternatively, self-adaptive models (e.g. ANNs) can be employed to address this issue. Some selected solutions can be tested by an accurate model and then used to retrain the ANNs during the search. By doing so, the prediction quality of the ANNs will keep improving and therefore reduce the search bias.

Another possible use of the ANNs might be to sample the objective space. Let us assume a good solution has been found, we can also find a set of neighbour solutions by using gradient descent or ascent methods in which the objective function decreases or increases most - if the objective function is differentiable. It might be difficult to do this with a complicated model but if an ANN is trained as a surrogate, which is differentiable, this can be done.

A typical three-layer feed forward neural network, as used in this work, can be described as:

$$y = \varphi(w^h \cdot \varphi(w^i \cdot x)) \tag{8.1}$$

where y is the output of the network or the predicted objective function; φ is a differentiable non-linear transfer function; w^h and w^i are the hidden and input

weights vector, respectively; x is an input variable vector, in this case, an LP (but the actual values are replaced by the stand-alone K_{eff} or PPF. Please refer to Appendix B and Jiang et al. (2008)). The gradient of the given ANN can be derived:

$$\frac{dy}{dx} = \varphi^{'}(w^h \cdot \varphi(w^i \cdot x)) \cdot w^h \cdot \varphi^{'}(w^i \cdot x) \cdot w^i \tag{8.2}$$

Given:

$$\varphi(x) = \frac{1}{1 + e^{-x}} \tag{8.3}$$

$\varphi^{'}$ is:

$$\varphi^{'}(x) = \frac{-e^{-x}}{(1 + e^{-x}) \cdot (1 + e^{-x})} \tag{8.4}$$

To find new solutions that increase or decrease the objective function, new x is:

$$x_{t+1} = x_t + a \cdot \frac{dy}{dx} \qquad \text{or} \qquad x_{t+1} = x_t - a \cdot \frac{dy}{dx} \tag{8.5}$$

respectively, where a is small real number or the step size.

We start from a solution with its $K_{eff} = 1.04625$ and $PPF = 1.15826$, apply the above method to the ANNs for K_{eff} and PPF for 100 steps, with a step size of 0.01 and 0.001, respectively. Each x_{t+1} is tested with the ANN again. The results are shown in figure 8.1.

The newly generated x vectors by using the gradient of the ANNs appear to be maximising/minimising the estimated K_{eff} and PPF to a local optimum, However, since they are direct input to the ANNs, and consist of real values (Appendix B), they cannot be accurately mapped back to valid LPs. Figure 8.1 only help to illustrate the local shape of the approximated objective functions. How to use this information in finding optimal solutions in decision variable space has yet to be investigated.

Another use of the gradient of the ANN is in sensitivity analysis. As each input variable for the ANNs represents a fuel channel, a large gradient at channel i, $\frac{\partial ANN}{\partial x_i}$, suggests that a change at this position may result in significant change

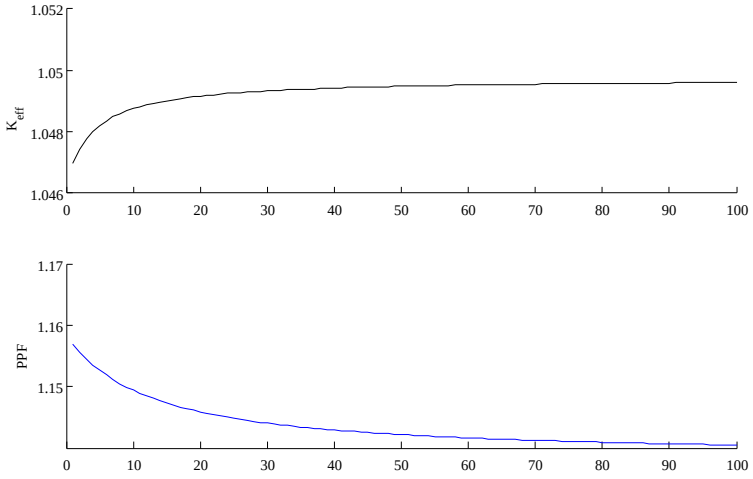


Figure 8.1: Using the gradient of the ANNs to estimate local distribution of K_{eff} and PPF

in objective space, and therefore more candidate LPs should be tested by changing this loading position.

The use of the ANN based surrogate model in loading pattern optimisation merits further investigation. As discussed above, it might be very useful in the analysis of both objective space and decision variable space, and possibly in the improvement of the optimisation algorithms.

8.3.3 Hybrid Algorithms

Although not new, hybridising the EDAs with other optimisation techniques and/or heuristic information can improve optimisation performance and can be used as a more practical optimisation tool, e.g. applying n-opt local searcher to the current best at the end of each generation or the end of the EDA search.

References

D. H. Ahn and S. H. Levine. Automatic optimized reload and depletion method for a pressurized water reactor. *Nuclear Technology*, 71:535–547, 1985.

R. Avery. Theory of coupled reactors. In *2nd UN Conference on Peaceful Uses of Atomic Energy*, Geneva, 1958.

T. Back, D. B. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. Oxford University Press and the Institute of Physics, 1997.

J. E. Baker. Adaptive selection methods for genetic algorithms. In *Proceedings of the first International Conference on Genetic Algorithms and their Applications*, pages 101–111, 1985.

K. Balakrishnan and A. Kakodkar. Optimization of the initial fuel loading of the indian PHWR with thorium bundles for achieving full power. *Annals of Nuclear Energy*, 21(1):1–9, 1993.

S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. CMU-CS-94-163, Carnegie Mellon University, 1994.

D. Beasley, D. R. Bull, and R. R. Martin. An overview of genetic algorithms: Part 2, research topics. *University Computing*, 15(4):170–181, 1993. URL `citeseer.ist.psu.edu/article/beasley93overview.html`.

J. S. De Bonet, C. Isbell, and P. Viola. MIMIC: Finding optima by estimating probability densities. In *Advances in Neural Information Processing*, volume 9, Denver 1996, 1996. MIT Press, Cambridge.

P. A. N. Bosman and D. Thierens. Exploiting gradient information in continous iterated density estimation evolutionary algorithms. In *Proceedings of the BNAIC-2001 thirteenth Belgium-Netherlands Conference on Artificial Intelligence*, pages 69–76, 2001.

E. S. Buffa, G. C. Armour, and T. E. Vollmann. Allocating facilities with craft. *Harvard Business Review*, 42:136–158, 1964.

A. Castillo, G. Alonso, L. B. Morales, C. M. del Campo, J. L. Francois, and E. del Valle. BWR fuel reload design using a tabu search technique. *Annals of Nuclear Energy*, 31:151–161, 2004.

J. L. C. Chapot, F. C. D. Silva, and R. Schirru. A new approach to the use of genetic algorithms to solve the pressurized water reactor's fuel management optimisation problem. *Annals of Nuclear Energy*, 26:641–655, 1999.

K. A. de Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.

K. A. de Jong and W. M. Spears. Using genetic algorithms to solve p-complete problems. In *Proc. of Third Int. Conf. Genetic Algorithms and their Applications*, pages 124–132. Kaufmann, 1989.

C. R. E. de Oliveira, M. D. Eaton, A. P. Umpleby, and C. C. Pain. Finite element spherical harmonics solutions of the 3d Kobayashi benchmarks with ray-tracing void treatment. *Prog. Nuclear Energy*, 261(39):243–261, 2001.

K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation*, 6:182–197, 2002.

M. D. DeChaine and M. A. Feltus. Fuel management optimisation using genetic algorithms and expert system. *Nuclear Science and Engineering*, 124:188–196, 1996.

C. Martin del Campo and J. L. Francois. Boiling water reactor fuel assembly axial design optimization using tabu search. *Nuclear Science and Engineering*, 142: 107–115, 2002.

M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to traveling salesman problem. *IEEE Transactions on Evolutionary Computations*, 1(1):53–66, 1997.

A. Erdogan and M. Geckinli. A PWR reload optimisation code (XCORE) using artificial neural networks and genetic algorithms. *Annals of Nuclear Energy*, 30: 35–53, 2003.

R. Etxeberria and P. Larrañaga. Global optimisation using Bayesian networks. In *The Second Symposium on Artificial Intelligence. CIMAF99. Special session on Distributions and Evolutionary Optimisation*, pages 332–339, 1999.

E. F. Faria and C. Pereira. Nuclear fuel loading pattern optimisation using a neural network. *Annals of Nuclear Energy*, 30:603–613, 2003.

B. R. Fox and M. B. McMahon. Genetic operators for sequencing problems. In *Foundations of Genetic Algorithms*, pages 284–300. Kaufmann, 1990.

S. J. Franklin, A. J. H. Goddard, and J. S. O'Connell. Research reactor facilities and recent developments at Imperial College, London. In *Research Reactor Fuel Management 1998: European Nuclear Society*, 1998.

S. J. Franklin, D. Gardner, J. Mumford, R. Lea, and J. Knight. Business operations and decommissioning strategy for Imperial College London research reactor 'CONSORT' - a financial risk management approach. In *Research Reactor Fuel Management 2005, European Nuclear Society*, 2005.

C. Fyfe. Structured population-based incremental learning. *Soft Computing*, 2: 191–198, 1999.

A. Galperin and Y. Kimhy. Application of knowledge-based methods to in-core fuel management. *Nuclear Science and Engineering*, 109:103–110, 1991.

Y. Gao and J. Culberson. Space complexity of estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 13(1):125–143, 2005.

D. E. Goldberg. Zen and the art of genetic algorithms. In *Proceedings of the third International Conference on Genetic Algorithms*, pages 80–85, George Mason

University, USA, 1989a.

D. E. Goldberg. *Genetic Algorithms in Search, Optimisation and Machine Learning.* Addison Wesley, 1989b.

C. Gonzalez, J. A. Lozano, and P. Larrañaga. Analyzing the PBIL algorithm by means of discrete dynamical systems. *Complex Systems*, 12:465–479, 2000.

J. Grefenstette, R. Gopal, B. Rosmaita, and D. Van Gucht. Genetic algorithms for the traveling salesman problem. In *Proceedings of the first International Conference on Genetic Algorithms and their Applications*, page 160, 1985.

G. R. Harik, F. G. Lobo, and D. E. Goldberg. The compact genetic algorithm. *IEEE Transactions on Evolutionary Computation*, 3:287–297, 1999.

I. A. Ben Hmaida, J. N. Carter, C. R. E. de Oliveira, A. J. H. Goddard, and G. T. Parks. Nuclear in-core fuel management optimisation using the tabu search method. In *Int Conf of Maths and Comp in Nuclear Apps*, 1999.

J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

X. Huo and Z. Xie. A novel channel selection method for CANDU refuelling based on the BPANN and GA techniques. *Annals of Nuclear Energy*, 32:1081–1099, 2005.

S. Jiang. A genetic local search algorithm for the satellite broadcasting scheduling problems. Master's thesis, University of Essex, 2003.

S. Jiang, A. K. Ziver, J. N. Carter, C. C. Pain, A. J. H. Goddard, S. J. Franklin, and H. Phillips. Estimation of distribution algorithms for nuclear reactor fuel management optimisation. *Annals of Nuclear Energy*, 33(12):1039–1057, 2006.

S. Jiang, C. C. Pain, J. N. Carter, A. K. Ziver, M. D. Eaton, A. J. H. Goddard, S. J. Franklin, and H. J. Phillips. Nuclear reactor reactivity prediction using feed forward artificial neural networks. In *Advances in Neural Networks - ISNN 2008*, volume 5263/2008, pages 400–409. Springer Berlin / Heidelberg, 2008.

T. K. Kim and C. H. Kim. Mixed integer programming for pressurized water reactor

fuel-loading-pattern optimization. *Nuclear Science and Engineering*, 127:346–357, 1997.

J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

D. J. Kropaczek and P. J. Turinsky. In-core nuclear fuel management optimization for pressurized water reactors utilizing simulated annealing. *Nuclear Technology*, 95:9–31, 1991.

D. J. Kropaczek, G. T. Parks, G. I. Maldonano, and P. J. Turinsky. Application of simulated annealing to in-core nuclear fuel management optimisation. In *Proc. Int. Topl. Mtg. Advances in Mathematics,Computations and Reactor Physics*, volume 5, page 22.1 1.1, 1991.

P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms A New Tool for Evolutionary Computation*. Kluwer Academic Press, Boston, 2001.

H. Li and Q. Zhang. Hybrid estimation of distribution algorithm for multi-objective knapsack problem. In *Proceedings of the 4th European Conference on Evolutionary Computation in Combinatorial Optimization*, April 2004.

H. Li and Q. Zhang. Multiobjective optimisation problems with compoicated Pareto sets, MOEA/D and NSGA-II. *IEEE Trans. on Evolutionary Computation*, 12(2): 284–302, 2009.

C. Lin, J. Yang, K. Lin, and Z. Wang. Pressurized water reactor loading pattern design using the simple tabu search. *Nuclear Science and Engineering*, 129:61–71, 1998.

F. Lin, C. Kao, and C. Hsu. Applying the genetic approach to simulated annealing in solving some np-hard problems. *IEEE Transactions on Systems, Man. and Cybernetics*, 23(6):1752–1767, 1993.

L. Machado and R. Schirru. The ant-q algorithm applied to the nuclear reload problem. *Annals of Nuclear Energy*, 29:1455–1470, 2002.

Y. P. Mahlers. Core loading pattern optimization for research reactors. *Annals of*

Nuclear Energy, 24:509–514, 1997.

M. Melice. Pressurized water reactor optimal core management and reactivity profiles. *Nuclear Science and Engineering*, 37:451, 1969.

Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1997.

K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1999.

J. S. Miller and N. D. Eckhoff. A linear programming fuel management model for the HTGR. *Annals of Nuclear Energy*, 2:649–656, 1975.

T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.

H. Müehlenbein. The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5:303–346, 1998.

H. Müehlenbein and R. Hons. A theoretical and experimental investigation of estimation of distribution algorithms. In *Genetic and Evolutionary Computation Conference (GECCO) 2004*, 2004.

H. Mühlenbein. The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5:303–346, 1998.

H. Mühlenbein and T. Mahnig. Convergence theory and application of the factorized distribution algorithm. *Journal of Computing and Information Technology*, 7:19–32, 1999.

H. Mühlenbein and G. Paab. From recombination of genes to the estimation of distributions i. binary parameters. *Lecture Notes In Computer Science*, 1141: 178–187, 1996.

H. Mühlenbein, T. Mahnig, and A. O. Rodriguez. Schemata, distributions, and graphical models in evolutionary optimization. *Journal of Heuristics*, 5:215–247, 1999.

US Nuclear Regulatory Commission NRC. Shutdown margin, January 2008.

I. M. Oliver, D. J. Smith, and J. R. C. Holland. A study of permutation crossover

operators on the travelling salesman problem. In *Proceedings of the 2nd International Conference of Genetic Algorithms and Their Applications*, pages 224–230, 1987.

J. J. Ortiz and I. Requena. Using a multi-state recurrent neural network to optimize loading patterns in BWRs. *Annals of Nuclear Energ*, 31:789–803, 2004.

C. Pain, A. J. H. Goddard, and C. R. E. de Oliveira. Gradient annealing: An approach to combinatorial optimisation. Technical report, Imperial College London, 1995.

G. T. Parks. Optimization of advanced gas-cooled reactor fuel performance by a stochastic method. *Nuclear Energy*, 26:319–327, 1987.

G. T. Parks. An intelligent stochastic optimization routine for nuclear fuel cycle design. *Nuclear Technology*, 89:233–246, 1990.

G. T. Parks. Multiobjective pressurized water reactor reload core design by nondominated genetic algorithm search. *Nuclear Science and Engineering*, 124:178–187, 1996.

G. T. Parks. Multiobjective pressurised water reactor reload core design using a genetic algorithm. In *3rd International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA'97), Norwich, 1, 53-57*, volume 1, pages 52–57, 1997.

J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

M. Pelikan and H. Mühlenbein. The bivariate marginal distribution algorithm. In R. Roy, T. Furuhashi, and P. K. Chawdhry, editors, *Advances in Soft Computing - Engineering Design and Manufacturing*, pages 521–535, London, 1999. Springer-Verlag. ISBN 1-85233-062-7.

C. M. N. A. Pereira and C. M. F. Lapa. Coarse-grained parallel genetic algorithm applied to a nuclear reactor core design optimization problem. *Annals of Nuclear Energy*, 30:555–565, 2003.

P. W. Poon. *The Genetic Algorithm Applied to PWR Reload Core Design*. PhD thesis, Department of Engineering, University of Cambridge, 1992.

P. W. Poon and J. N. Carter. Genetic algorithm crossover operators for ordering applications. *Computers and Operations Research*, 22(1):135–147, 1995.

P. W. Poon and G. T. Parks. Application of genetic algorithms to in-core nuclear fuel management optimization. In *Proc. Joint Int. Conf. Mathematical Methods and Super-computing in Nuclear Applications*, volume 2, pages 777–786, 1993.

K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach To Global Optimization*. Springer, 2005.

A. J. Quist, R. van Geemert, J. E. Hoogenboom, T. Illes, C. Roos, and T. Terlaky. Application of nonlinear optimization to reactor core fuel reloading. *Annals of Nuclear Energy*, 26:423–448, 1999.

I. Rechenberg. *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart, 1973.

G. Reinelt. TSPLIB: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html, 1997. Internet Resource, last visited in Nov.2007.

S. Rudlof and M. Koppen. Stochastic hill climbing by vectors of normal distributions. In *Proceedings of the First Online Workshop on Soft Computing*, 1996.

M. Sadighi, S. Setayeshi, and A. A. Salhi. PWR fuel management optimization using neural networks. *Annals of Nuclear Energy*, 29:41–51, 2002.

T. O. Sauar. Application of linear programming to in-core fuel management optimization in light water reactors. *Nuclear Science and Engineering*, 46:274–283, 1971.

SERCO-ANSWERS. *ANSWERS/WIMS User Guide for Version 8*, 1999.

C. E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. Journal*, pages 379–423 and 623–656, 1948.

T. Smuc, D. Pevec, and B. Petrovic. Annealing strategies for loading pattern optimization. *Annals of Nuclear Energy*, 21:325–336, 1994.

J. A. Snyman. *Practical Mathematical Optimization: An Introduction to basic optimization theory and classical and new gradient-based algorithms.* New York:Springer, 2005.

J. G. Stevens, K. S. Smith, and K. R. Rempe. Optimization of pressurized water reactor shuffling by simulated annealing with heuristics. *Nuclear Science and Engineering*, 121:67–88, 1995.

J. S. Suh and S. H. Levine. Optimized automatic reload program for pressurized water reactors using simple direct optimization techniques. *Nuclear Science and Engineering*, 105:371–382, 1990.

The Paks Nuclear Power Plant Company. VVER440/213 - the reactor core, 2007. Internet Resource, last visited in Jan. 2008.

V. G. Toshinsky, H. Sekimoto, and G. I. Toshinsky. A method to improve multi-objective genetic algorithm optimisation of a self-fuel-providing LMFBR by niche induction among nondominated solutions. *Annals of Nuclear Energy*, 27:397–410, 2000.

S. Tsutsui, M. Pelikan, and D. E. Goldberg. Evolutionary algorithm using marginal histogram models in continuous domain. Illigal report 2001019, University of Illinois at Urbana-Champaign, 2001.

P. J. Turinsky and G. T. Parks. *Advances in Nuclear Science and Technology*, chapter 6, pages 137–167. Springer, 1999.

P. J. Turinsky, P. M. Keller, and H . S. Abdel-khalik. Evolution of nuclear fuel management and reactor operational aid tools. *Nuclear Engineering and Technology*, 37:79–90, 2005.

R. van Geemert, A. J. Quist, J. E. Hoogenboom, and H. P. M. Gibcus. Research reactor in-core fuel management optimisation by application of multiple cyclic interchange algorithms. *Nuclear Engineering and Design*, 186:369–377, 1998.

I. Wall and H. Fenech. The application of dynamic programming to fuel management optimization. *Nuclear Science and Engineering*, 22:285–297, 1965.

D. Whitley. The genitor algorithm and selection pressure: why rank-based allocation of reproductive trials is best. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 116–121, Arlington, VA, 1989.

D. Whitley, T. Starkweather, and D. Fuquay. Scheduling problems and the traveling salesman: the genetic edge recombination operator. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 116–121, Arlington, VA, 1989.

World Nuclear Association WNA. Nuclear power in the world today, August 2007. Internet Resource, last visited in Jan. 2008.

A. Zell, G. Mauier, M. Vogt, and N. Mache. *SNNS Stuttgart Neural Network Simulator User Manual Version 4.1.*, 1995.

Q. Zhang. CC483 Evolutionary Computation: Lecture notes, Department of Computer Science, University of Essex, 2003.

Q. Zhang. On the convergence of a factorized distribution algorithm with truncation selection. *Complexity*, 9(4):17–23, 2004a.

Q. Zhang. On stability of fixed points of limit models of univariate marginal distribution algorithm and factorized distribution algorithm. *IEEE Trans. on Evolutionary Computation*, 8(1):80–93, 2004b.

Q. Zhang and H. Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. on Evolutionary Computation*, 11(6):712–731, 2007.

Q. Zhang and H. Müehlenbein. On the convergence of a class of estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 8(2): 127–136, 2004.

Q. Zhang, J. Sun, E. Tsang, and J. Ford. Hybrid estimation of distribution algorithm for global optimisation. *Engineering Computations*, 21(1):91–107, 2004.

Q. Zhang, J. Sun, and E.P.K. Tsang. Evolutionary algorithm with the guided mutation for the maximum clique problem. *IEEE Transactions on Evolutionary Computation*, 9(2):192–200, 2005.

Q. Zhang, J. Sun, E. Tsang, and J. Ford. *Towards a New Evolutionary Computation : Advances in the Estimation of Distribution Algorithms*, chapter 12, pages 281–292. Springer Berlin / Heidelberg, 2006.

Q. Zhang, A. Zhou, and Y. Jin. RM-MEDA: A regularity model based multiobjective estimation of distribution algorithm. *IEEE Trans. on Evolutionary Computation*, 12(1):41–63, 2008.

J. Zhao, B. Knight, E. Nissan, and A. Soper. Fuelgen: a genetic algorithm-based system for fuel loading pattern design in nuclear power reactors. *Expert System with Applications*, 14:461–470, 1998.

A. K. Ziver. Private communication, 2005.

A. K. Ziver, C. C. Pain, J. N. Carter, C. R. E. de Oliveira, A. J. H. Goddard, and R. S. Overton. Genetic algorithms and artificial neural networks for loading pattern optimisation of advanced gas-cooled reactors. *Annals of Nuclear Energy*, 31:431–457, 2004.

Appendix A

Coupled Reactor Theory and the calculation of the Stand-alone K_{eff} for the CONSORT reactor

The theory of stand-alone K_{eff} with fuel coupling can be explained using the coupled reactor theory (Avery 1958). In the CONSORT research reactor there are 24 channels in which fuel elements can be inserted. In order to outline the coupled reactor theory each fuel element in the core is considered as a separate reactor. Thus the CONSORT reactor is composed of 24 fissile zones that are coupled together to form the whole reactor. Based on this representation one can write an expression for the total fission sources as follows by decoupling the reactor into 24 separate reactors:

$$S_i = \sum_{j=1}^{24} S_{ij}, \;\; i = 1, ...n \tag{A.1}$$

here the subscript j denotes each fuel element from 1 to 24 and off-diagonal terms, for example S_{12} gives the fission source in reactor 1 due to neutrons born in reactor 2. The diagonal terms S_{11},S_{22} . . . give the fission sources due to neutrons born in the same reactor, in other words fission sources due to each fuel element (reactor) are stand-alone. These partial source terms can be defined as:

$$S_i = \int \int (v\Sigma_f)_i \Phi_j dr dE \tag{A.2}$$

where $(v\sum_f)_i$ is the macroscopic production cross-section in reactor i; Φ_j is the scalar forward flux distribution in reactor j, dr the integration over space, the volume element, and dE is the integration over neutron energies. Using the above notation, the coupling coefficients can be defined as:

$$k_{jn} = \frac{S_{jn}}{S_k} \tag{A.3}$$

So, when $j = i$ (i.e. for diagonal terms) the stand-alone K_{eff} can be written as:

$$K_{eff} = k_{ij} \tag{A.4}$$

Introducing the coupling coefficients in Eq. A.1 the following equations can be derived:

$$S_i = \sum_{j=1}^{24} S_{ij} \cdot k_{ij},\ i = 1, ...n \tag{A.5}$$

Although the EDA approach does not require any problem-specific information, the use of sensible heuristics can improve the optimisation and speed up the convergence. In this case, we introduce the stand-alone K_{eff} with fuel coupling as useful heuristic information that can be used in LP optimisation.

When LPs are being optimised, the fuel types, the available positions of the fuel channels in the core and the effect of different fuel type coupling should be considered. The infinite multiplication factor indicates the reactivity that can be achieved by fuel assemblies in a non-leakage environment, and is therefore used in LP optimization to define different types of fuel elements. But it does not contain any channel dependent (reactor core position) information and also ignores the interaction between adjacent fuel assemblies. In order to include the effect of channel positions and 'fuel coupling' in optimisation, position-dependent and neighborhood-

fuel-dependent fuel assembly information is required. We make use of an idea proposed in Erdogan and Geckinli (2003) that provides the position-dependent fuel measurement and extends it to include the effect from adjacent fuel assemblies.

The idea is to load only one fuel channel with a specified fuel assembly and fill the other channels with water, then calculate the reactor core 'K_{eff}'. This is repeated for all channels and fuel types. For example, if we have 24 fuel channels and 5 fuel types then only 120 LPs need to be evaluated. The resulting 24x5 matrix is called the stand-alone K_{eff} without fuel coupling. Erdogan and Geckinli have used this to build an Artificial Neural Network (ANN) predicting the K_{eff} and maximum power fraction. However, they did not make use of this information in their GA for LP optimisation.

In fact, this 'stand-alone' K_{eff} can be used in optimisation algorithms as heuristic information H, as described in a previous chapter, because it contains both information from the fuel assembly's K_{∞} (reactivity) and the effect of fuel channel position. The stand-alone K_{eff} is a position-dependent measurement of fuel types.

However, the stand-alone K_{eff} doesn't contain the fuel 'coupling' information. According to the coupled reactor theory Avery (1958), we can calculate the stand-alone K_{eff} in a more sensible way. For example, insert fuel type j at channel i, and fill all other channels with fuel type m instead of water. In this work, we calculate the stand-alone K_{eff} coupled with MARK I_A because it has the median K_{∞}. So MARK I_A can be regarded as a 'generic' fuel type among the five types. Therefore the resulting 'stand-alone K_{eff} coupling with fuel MARK I_A' contains the physical character of fuel j, the effect of fuel channel i and the effect of coupling fuel j with a generic fuel element - MARK I_A, in this case.

The calculated stand-alone K_{eff}s coupled with fuel MARK I_A (Test Case 1) are listed in table 7.3. The table entry (i, j) is the calculated K_{eff} when channel i loaded with fuel type j and other channels filled with MARK I_A.

The stand-alone K_{eff} values were calculated using the EVENT radiation trans-

port code for the CONSORT reactor, for both test cases in the main text. For each case the standard deviation was also calculated, using the equation:

$$\delta_j = \sqrt{\frac{\sum_{i=1}^{n} \left(k_{ij} - \frac{1}{n} \cdot \sum_{i=1}^{n} k_{ij}\right)^2}{n-1}} \tag{A.6}$$

where i and j are the indices of fuel channels and fuel types, respectively; $\frac{1}{n} \cdot \sum_{i=1}^{n} k_{ij}$ is the mean of the calculated K_{eff} with fuel type j at channel i and other channels filled with a generic fuel (e.g., MARK I_A in Test Case 1 in the main text). The constant n is the number of fuel channels.

The calculated stand-alone K_{eff}s are listed in tables A.1 to A.7 and A.8 to A.14 for Test Case 2 and 3, respectively. Tables A.15 to A.21 shows the stand-alone PPF for Test Case 3.

It can be seen that the variations of K_{eff} and PPF for each type of fuel from position 1 to 24 are small in absolute value. But this variation was found to be very important for training the ANN to predict the overall K_{eff} and PPF of the reactor core, as well as for optimising loading patterns.

Channel No.	Fuel 1	Fuel 2	Fuel 3	Fuel 4	Fuel 5
1	1.009690	1.009690	1.009670	1.009620	1.009570
2	1.008760	1.008760	1.008740	1.008690	1.008650
3	1.008970	1.008960	1.008940	1.008910	1.008870
4	1.009030	1.009030	1.009010	1.008980	1.008960
5	1.009200	1.009200	1.009190	1.009170	1.009150
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	1.009370	1.009370	1.009310	1.009190	1.009080
8	1.009620	1.009620	1.009580	1.009490	1.009420
9	1.008880	1.008880	1.008850	1.008790	1.008730
10	1.009320	1.009320	1.009300	1.009260	1.009230
11	1.008760	1.008760	1.008720	1.008640	1.008570
12	1.009170	1.009170	1.009150	1.009090	1.009050
13	1.009070	1.009070	1.009000	1.008840	1.008710
14	1.009060	1.009050	1.009000	1.008890	1.008790
15	0.000000	0.000000	0.000000	0.000000	0.000000
16	1.008490	1.008490	1.008460	1.008400	1.008350
17	1.008710	1.008710	1.008690	1.008650	1.008610
18	1.008340	1.008340	1.008290	1.008190	1.008100
19	1.008550	1.008550	1.008510	1.008420	1.008340
20	1.009670	1.009660	1.009600	1.009470	1.009350
21	1.009040	1.009040	1.009030	1.009000	1.008970
22	1.009130	1.009130	1.009110	1.009090	1.009070
23	1.009050	1.009040	1.009020	1.008960	1.008910
24	1.009140	1.009140	1.009110	1.009060	1.009010

Table A.1: The calculated stand-alone K_{eff}s using EVENT for Test Case 2 with a fuel inventory of 35 fuel elements in total, Part I

Channel No.	Fuel 6	Fuel 7	Fuel 8	Fuel 9	Fuel 10
1	1.009400	0.996383	1.009360	1.009360	0.000000
2	1.008480	0.995316	1.008450	1.008450	0.000000
3	1.008730	0.997963	1.008710	1.008700	0.000000
4	1.008850	0.999873	1.008830	1.008830	0.000000
5	1.009070	1.002710	1.009060	1.009060	0.000000
6	0.000000	0.000000	0.000000	0.000000	1.009590
7	1.008650	0.980490	1.008570	1.008560	0.000000
8	1.009120	0.990179	1.009070	1.009060	0.000000
9	1.008520	0.991832	1.008480	1.008470	0.000000
10	1.009080	0.998556	1.009060	1.009050	0.000000
11	1.008300	0.987489	1.008260	1.008250	0.000000
12	1.008860	0.995060	1.008830	1.008820	0.000000
13	1.008170	0.972006	1.008080	1.008060	0.000000
14	1.008400	0.981803	1.008330	1.008320	0.000000
15	0.000000	0.000000	0.000000	0.000000	0.000000
16	1.008140	0.991718	1.008110	1.008100	0.000000
17	1.008450	0.995940	1.008420	1.008410	0.000000
18	1.007740	0.982752	1.007680	1.007670	0.000000
19	1.008030	0.986738	1.007970	1.007960	0.000000
20	1.008900	0.977475	1.008820	1.008810	0.000000
21	1.008860	0.999680	1.008840	1.008840	0.000000
22	1.008980	1.001380	1.008960	1.008960	0.000000
23	1.008710	0.992861	1.008670	1.008670	0.000000
24	1.008830	0.994637	1.008800	1.008790	0.000000

Table A.2: The calculated stand-alone K_{eff}s using EVENT for Test Case 2 with a fuel inventory of 35 fuel elements in total, Part II

Channel No.	Fuel 11	Fuel 12	Fuel 13	Fuel 14	Fuel 15
1	0.000000	1.009990	1.009940	1.009740	1.009710
2	0.000000	1.009990	1.009940	1.009740	1.009710
3	0.000000	1.009920	1.009880	1.009720	1.009690
4	0.000000	1.009830	1.009800	1.009680	1.009660
5	0.000000	1.009760	1.009740	1.009650	1.009640
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	0.000000	1.010610	1.010470	1.009980	1.009890
8	0.000000	1.010290	1.010190	1.009860	1.009800
9	0.000000	1.010100	1.010030	1.009780	1.009740
10	0.000000	1.009920	1.009880	1.009720	1.009690
11	0.000000	1.010220	1.010140	1.009830	1.009780
12	0.000000	1.010020	1.009960	1.009750	1.009720
13	0.000000	1.010860	1.010680	1.010070	1.009970
14	0.000000	1.010510	1.010380	1.009940	1.009870
15	1.009590	0.000000	0.000000	0.000000	0.000000
16	0.000000	1.010080	1.010010	1.009780	1.009730
17	0.000000	1.009970	1.009910	1.009730	1.009700
18	0.000000	1.010450	1.010330	1.009920	1.009850
19	0.000000	1.010340	1.010230	1.009870	1.009810
20	0.000000	1.010650	1.010500	1.009990	1.009910
21	0.000000	1.009830	1.009800	1.009680	1.009660
22	0.000000	1.009790	1.009760	1.009670	1.009650
23	0.000000	1.010050	1.009990	1.009770	1.009730
24	0.000000	1.010010	1.009950	1.009750	1.009710

Table A.3: The calculated stand-alone K_{eff}s using EVENT for Test Case 2 with a fuel inventory of 35 fuel elements in total, Part III

Channel No.	Fuel 16	Fuel 17	Fuel 18	Fuel 19	Fuel 20
1	1.009650	1.009590	1.009590	1.009580	1.009570
2	1.009650	1.009590	1.009590	1.009580	1.009570
3	1.009640	1.009590	1.009590	1.009580	1.009580
4	1.009620	1.009590	1.009590	1.009580	1.009580
5	1.009610	1.009590	1.009590	1.009580	1.009580
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	1.009740	1.009600	1.009590	1.009560	1.009550
8	1.009690	1.009600	1.009590	1.009570	1.009560
9	1.009670	1.009600	1.009590	1.009570	1.009570
10	1.009640	1.009590	1.009590	1.009580	1.009580
11	1.009680	1.009600	1.009590	1.009570	1.009560
12	1.009650	1.009590	1.009590	1.009580	1.009570
13	1.009780	1.009600	1.009590	1.009550	1.009540
14	1.009730	1.009600	1.009590	1.009560	1.009550
15	0.000000	0.000000	0.000000	0.000000	0.000000
16	1.009660	1.009600	1.009590	1.009570	1.009570
17	1.009640	1.009590	1.009590	1.009580	1.009570
18	1.009720	1.009600	1.009590	1.009560	1.009550
19	1.009700	1.009600	1.009590	1.009570	1.009560
20	1.009750	1.009600	1.009590	1.009560	1.009550
21	1.009620	1.009590	1.009590	1.009580	1.009580
22	1.009620	1.009590	1.009590	1.009580	1.009580
23	1.009660	1.009600	1.009590	1.009580	1.009570
24	1.009650	1.009590	1.009590	1.009580	1.009570

Table A.4: The calculated stand-alone K_{eff}s using EVENT for Test Case 2 with a fuel inventory of 35 fuel elements in total, Part IV

Channel No.	Fuel 21	Fuel 22	Fuel 23	Fuel 24	Fuel 25
1	1.009570	1.009520	1.009430	1.009420	1.009420
2	1.009570	1.009520	1.009440	1.009420	1.009420
3	1.009570	1.009530	1.009460	1.009450	1.009450
4	1.009580	1.009540	1.009500	1.009490	1.009490
5	1.009580	1.009560	1.009520	1.009520	1.009520
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	1.009540	1.009400	1.009200	1.009160	1.009150
8	1.009550	1.009460	1.009320	1.009300	1.009290
9	1.009560	1.009500	1.009400	1.009380	1.009370
10	1.009570	1.009530	1.009460	1.009450	1.009450
11	1.009560	1.009470	1.009350	1.009320	1.009320
12	1.009570	1.009510	1.009420	1.009410	1.009400
13	1.009520	1.009360	1.009100	1.009060	1.009050
14	1.009540	1.009420	1.009240	1.009200	1.009190
15	0.000000	0.000000	0.000000	0.000000	0.000000
16	1.009560	1.009500	1.009400	1.009380	1.009380
17	1.009570	1.009520	1.009440	1.009430	1.009430
18	1.009540	1.009430	1.009260	1.009230	1.009220
19	1.009550	1.009450	1.009300	1.009280	1.009270
20	1.009530	1.009390	1.009180	1.009150	1.009140
21	1.009580	1.009540	1.009500	1.009490	1.009490
22	1.009580	1.009550	1.009510	1.009500	1.009500
23	1.009560	1.009500	1.009410	1.009400	1.009390
24	1.009570	1.009510	1.009430	1.009410	1.009410

Table A.5: The calculated stand-alone K_{eff}s using EVENT for Test Case 2 with a fuel inventory of 35 fuel elements in total, Part V

Channel No.	Fuel 26	Fuel 27	Fuel 28	Fuel 29	Fuel 30
1	1.009420	1.009400	1.009400	1.009400	1.009390
2	1.009420	1.009410	1.009400	1.009400	1.009390
3	1.009450	1.009440	1.009430	1.009430	1.009430
4	1.009490	1.009480	1.009480	1.009480	1.009470
5	1.009520	1.009510	1.009510	1.009510	1.009500
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	1.009150	1.009120	1.009110	1.009110	1.009090
8	1.009290	1.009270	1.009260	1.009260	1.009250
9	1.009370	1.009360	1.009350	1.009350	1.009340
10	1.009450	1.009440	1.009430	1.009430	1.009430
11	1.009320	1.009300	1.009290	1.009290	1.009280
12	1.009400	1.009390	1.009390	1.009390	1.009380
13	1.009050	1.009010	1.009000	1.009000	1.008970
14	1.009190	1.009170	1.009160	1.009160	1.009140
15	0.000000	0.000000	0.000000	0.000000	0.000000
16	1.009380	1.009370	1.009360	1.009360	1.009350
17	1.009430	1.009420	1.009410	1.009410	1.009400
18	1.009220	1.009200	1.009190	1.009190	1.009170
19	1.009270	1.009250	1.009240	1.009240	1.009220
20	1.009140	1.009100	1.009090	1.009090	1.009070
21	1.009490	1.009480	1.009480	1.009470	1.009470
22	1.009500	1.009500	1.009490	1.009490	1.009490
23	1.009390	1.009380	1.009370	1.009370	1.009360
24	1.009410	1.009400	1.009390	1.009390	1.009380

Table A.6: The calculated stand-alone K_{eff}s using EVENT for Test Case 2 with a fuel inventory of 35 fuel elements in total, Part VI

Channel No.	Fuel 31	Fuel 32	Fuel 33	Fuel 34	Fuel 35
1	1.009380	1.009370	1.009330	1.006090	1.002100
2	1.009380	1.009370	1.009330	1.007300	1.003030
3	1.009420	1.009410	1.009370	1.007600	1.004080
4	1.009470	1.009460	1.009430	1.008610	1.006280
5	1.009500	1.009490	1.009480	1.008860	1.007200
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	1.009070	1.009030	1.008920	1.000050	0.988367
8	1.009230	1.009200	1.009130	1.002920	0.994886
9	1.009330	1.009310	1.009260	1.006350	1.001120
10	1.009420	1.009410	1.009370	1.007180	1.003720
11	1.009260	1.009240	1.009170	1.005400	0.998760
12	1.009370	1.009350	1.009310	1.006540	1.002030
13	1.008940	1.008890	1.008760	0.997942	0.982654
14	1.009120	1.009080	1.008990	1.001640	0.990782
15	0.000000	0.000000	0.000000	0.000000	0.000000
16	1.009340	1.009320	1.009270	1.007060	1.001960
17	1.009400	1.009380	1.009340	1.007680	1.003780
18	1.009150	1.009110	1.009030	1.003000	0.992920
19	1.009210	1.009180	1.009100	1.003870	0.995300
20	1.009050	1.009010	1.008900	0.999564	0.987895
21	1.009460	1.009450	1.009430	1.008570	1.006230
22	1.009490	1.009480	1.009460	1.008740	1.006800
23	1.009350	1.009330	1.009290	1.006530	1.001820
24	1.009370	1.009360	1.009310	1.006760	1.002510

Table A.7: The calculated stand-alone K_{eff}s using EVENT for Test Case 2 with a fuel inventory of 35 fuel elements in total, Part VII

Channel No.	Fuel 1	Fuel 2	Fuel 3	Fuel 4	Fuel 5
1	1.081720	1.081380	1.081010	1.080010	1.080600
2	1.080010	1.079780	1.079540	1.078880	1.079270
3	1.080100	1.079910	1.079690	1.079110	1.079450
4	1.081320	1.081030	1.080710	1.079840	1.080350
5	1.081180	1.080940	1.080680	1.080000	1.080400
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	1.079420	1.078990	1.078550	1.077310	1.078040
8	1.080300	1.079940	1.079560	1.078510	1.079130
9	1.081770	1.081270	1.080740	1.079290	1.080150
10	1.081770	1.081380	1.080970	1.079830	1.080500
11	1.082120	1.081510	1.080870	1.079100	1.080140
12	1.082040	1.081560	1.081040	1.079640	1.080470
13	1.079070	1.078560	1.078030	1.076540	1.077420
14	1.079890	1.079440	1.078970	1.077690	1.078450
15	0.000000	0.000000	0.000000	0.000000	0.000000
16	1.081460	1.080990	1.080480	1.079090	1.079910
17	1.081280	1.080890	1.080470	1.079330	1.080000
18	1.078630	1.078310	1.077970	1.077030	1.077590
19	1.078840	1.078550	1.078250	1.077400	1.077900
20	1.080500	1.079980	1.079430	1.077920	1.078820
21	1.081520	1.081190	1.080850	1.079910	1.080460
22	1.081400	1.081110	1.080810	1.080000	1.080480
23	1.081190	1.080830	1.080450	1.079410	1.080030
24	1.081180	1.080850	1.080490	1.079520	1.080100

Table A.8: The calculated stand-alone K_{eff}s using EVENT for Test Case 3 with a fuel inventory of 35 fuel elements in total, Part I

Channel No.	Fuel 6	Fuel 7	Fuel 8	Fuel 9	Fuel 10
1	1.079370	1.078760	1.079730	1.078990	0.000000
2	1.078450	1.078040	1.078690	1.078200	0.000000
3	1.078730	1.078360	1.078940	1.078500	0.000000
4	1.079300	1.078770	1.079600	1.078970	0.000000
5	1.079570	1.079150	1.079810	1.079310	0.000000
6	0.000000	0.000000	0.000000	0.000000	1.080640
7	1.076510	1.075730	1.076960	1.076030	0.000000
8	1.077850	1.077200	1.078220	1.077450	0.000000
9	1.078370	1.077480	1.078880	1.077820	0.000000
10	1.079120	1.078430	1.079520	1.078700	0.000000
11	1.077980	1.076900	1.078610	1.077310	0.000000
12	1.078760	1.077910	1.079250	1.078240	0.000000
13	1.075590	1.074650	1.076120	1.075010	0.000000
14	1.076870	1.076070	1.077330	1.076380	0.000000
15	0.000000	0.000000	0.000000	0.000000	0.000000
16	1.078220	1.077370	1.078700	1.077700	0.000000
17	1.078610	1.077910	1.079010	1.078180	0.000000
18	1.076420	1.075830	1.076760	1.076060	0.000000
19	1.076850	1.076320	1.077160	1.076520	0.000000
20	1.076940	1.075980	1.077490	1.076350	0.000000
21	1.079320	1.078760	1.079650	1.078970	0.000000
22	1.079490	1.079000	1.079770	1.079180	0.000000
23	1.078750	1.078100	1.079120	1.078350	0.000000
24	1.078900	1.078290	1.079240	1.078520	0.000000

Table A.9: The calculated stand-alone K_{eff}s using EVENT for Test Case 3 with a fuel inventory of 35 fuel elements in total, Part II

Channel No.	Fuel 11	Fuel 12	Fuel 13	Fuel 14	Fuel 15
1	0.000000	1.080950	1.079160	1.080090	1.079740
2	0.000000	1.080860	1.079580	1.080250	1.080010
3	0.000000	1.080830	1.079710	1.080300	1.080080
4	0.000000	1.080900	1.079380	1.080170	1.079880
5	0.000000	1.080850	1.079640	1.080270	1.080030
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	0.000000	1.081050	1.078630	1.079910	1.079430
8	0.000000	1.080980	1.078970	1.080030	1.079630
9	0.000000	1.081090	1.078480	1.079840	1.079330
10	0.000000	1.080990	1.078960	1.080020	1.079620
11	0.000000	1.081190	1.078010	1.079670	1.079050
12	0.000000	1.081070	1.078570	1.079870	1.079380
13	0.000000	1.081130	1.078230	1.079760	1.079200
14	0.000000	1.081060	1.078580	1.079890	1.079400
15	1.080640	0.000000	0.000000	0.000000	0.000000
16	0.000000	1.081070	1.078570	1.079870	1.079390
17	0.000000	1.081000	1.078930	1.080010	1.079600
18	0.000000	1.080960	1.079090	1.080080	1.079710
19	0.000000	1.080930	1.079240	1.080130	1.079800
20	0.000000	1.081140	1.078190	1.079750	1.079170
21	0.000000	1.080920	1.079290	1.080140	1.079820
22	0.000000	1.080890	1.079470	1.080210	1.079930
23	0.000000	1.080970	1.079070	1.080060	1.079690
24	0.000000	1.080940	1.079170	1.080100	1.079750

Table A.10: The calculated stand-alone K_{eff}s using EVENT for Test Case 3 with a fuel inventory of 35 fuel elements in total, Part III

Channel No.	Fuel 16	Fuel 17	Fuel 18	Fuel 19	Fuel 20
1	1.078730	1.078970	1.080640	1.078740	1.078390
2	1.079270	1.079450	1.080640	1.079280	1.079030
3	1.079430	1.079590	1.080640	1.079440	1.079220
4	1.079020	1.079220	1.080640	1.079030	1.078730
5	1.079360	1.079510	1.080640	1.079360	1.079130
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	1.078040	1.078370	1.080640	1.078050	1.077570
8	1.078470	1.078750	1.080640	1.078490	1.078090
9	1.077850	1.078200	1.080640	1.077870	1.077370
10	1.078480	1.078750	1.080640	1.078490	1.078100
11	1.077250	1.077670	1.080640	1.077270	1.076660
12	1.077970	1.078300	1.080640	1.077980	1.077500
13	1.077520	1.077910	1.080640	1.077540	1.076950
14	1.077980	1.078310	1.080640	1.077990	1.077500
15	0.000000	0.000000	0.000000	0.000000	0.000000
16	1.077970	1.078300	1.080640	1.077980	1.077500
17	1.078440	1.078710	1.080640	1.078450	1.078050
18	1.078640	1.078890	1.080640	1.078650	1.078270
19	1.078830	1.079060	1.080640	1.078840	1.078510
20	1.077470	1.077870	1.080640	1.077490	1.076900
21	1.078900	1.079120	1.080640	1.078910	1.078600
22	1.079130	1.079320	1.080640	1.079140	1.078870
23	1.078610	1.078870	1.080640	1.078620	1.078250
24	1.078740	1.078980	1.080640	1.078760	1.078410

Table A.11: The calculated stand-alone K_{eff}s using EVENT for Test Case 3 with a fuel inventory of 35 fuel elements in total, Part IV

Channel No.	Fuel 21	Fuel 22	Fuel 23	Fuel 24	Fuel 25
1	1.077670	1.078080	1.078220	1.076600	1.076500
2	1.078500	1.078800	1.078910	1.077710	1.077640
3	1.078750	1.079020	1.079110	1.078060	1.077990
4	1.078130	1.078480	1.078600	1.077250	1.077170
5	1.078650	1.078930	1.079020	1.077960	1.077890
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	1.076550	1.077140	1.077330	1.075030	1.074890
8	1.077250	1.077730	1.077900	1.076010	1.075900
9	1.076320	1.076920	1.077120	1.074790	1.074650
10	1.077290	1.077760	1.077910	1.076110	1.076010
11	1.075380	1.076110	1.076360	1.073520	1.073350
12	1.076500	1.077080	1.077270	1.075040	1.074910
13	1.075730	1.076440	1.076670	1.073890	1.073720
14	1.076470	1.077060	1.077260	1.074930	1.074790
15	0.000000	0.000000	0.000000	0.000000	0.000000
16	1.076500	1.077080	1.077270	1.075040	1.074910
17	1.077230	1.077700	1.077870	1.076030	1.075920
18	1.077490	1.077940	1.078090	1.076320	1.076210
19	1.077800	1.078210	1.078350	1.076750	1.076650
20	1.075660	1.076370	1.076610	1.073790	1.073620
21	1.077960	1.078330	1.078450	1.077020	1.076930
22	1.078310	1.078630	1.078740	1.077500	1.077420
23	1.077470	1.077920	1.078070	1.076330	1.076220
24	1.077680	1.078100	1.078240	1.076610	1.076510

Table A.12: The calculated stand-alone K_{eff}s using EVENT for Test Case 3 with a fuel inventory of 35 fuel elements in total, Part V

Channel No.	Fuel 26	Fuel 27	Fuel 28	Fuel 29	Fuel 30
1	1.076030	1.077470	1.076250	1.077330	1.077100
2	1.077290	1.078350	1.077450	1.078250	1.078080
3	1.077680	1.078620	1.077830	1.078530	1.078380
4	1.076790	1.077970	1.076970	1.077850	1.077660
5	1.077590	1.078520	1.077730	1.078430	1.078280
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	1.074200	1.076270	1.074520	1.076070	1.075740
8	1.075340	1.077020	1.075600	1.076860	1.076590
9	1.073980	1.076040	1.074300	1.075840	1.075500
10	1.075490	1.077070	1.075730	1.076920	1.076660
11	1.072530	1.075030	1.072910	1.074790	1.074380
12	1.074270	1.076230	1.074570	1.076040	1.075720
13	1.072900	1.075390	1.073280	1.075160	1.074750
14	1.074100	1.076180	1.074420	1.075980	1.075640
15	0.000000	0.000000	0.000000	0.000000	0.000000
16	1.074270	1.076230	1.074570	1.076040	1.075720
17	1.075400	1.077010	1.075640	1.076850	1.076590
18	1.075690	1.077280	1.075930	1.077120	1.076870
19	1.076180	1.077610	1.076400	1.077470	1.077240
20	1.072780	1.075310	1.073170	1.075070	1.074660
21	1.076520	1.077780	1.076720	1.077660	1.077450
22	1.077070	1.078160	1.077230	1.078050	1.077870
23	1.075710	1.077260	1.075950	1.077110	1.076860
24	1.076040	1.077480	1.076260	1.077340	1.077110

Table A.13: The calculated stand-alone K_{eff}s using EVENT for Test Case 3 with a fuel inventory of 35 fuel elements in total, Part VI

Channel No.	Fuel 31	Fuel 32	Fuel 33	Fuel 34	Fuel 35
1	1.075830	1.076880	1.080360	1.080580	1.078060
2	1.077140	1.077920	1.080440	1.082980	1.078800
3	1.077550	1.078250	1.080470	1.082660	1.079010
4	1.076630	1.077490	1.080400	1.081100	1.078440
5	1.077460	1.078140	1.080450	1.081030	1.078890
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	1.073910	1.075440	1.080270	1.084400	1.077210
8	1.075100	1.076340	1.080330	1.082710	1.077790
9	1.073700	1.075200	1.080230	1.080990	1.076920
10	1.075270	1.076430	1.080320	1.080460	1.077760
11	1.072180	1.074010	1.080140	1.080860	1.076130
12	1.074000	1.075430	1.080240	1.080420	1.077080
13	1.072550	1.074390	1.080190	1.085240	1.076560
14	1.073800	1.075340	1.080250	1.083720	1.077130
15	0.000000	0.000000	0.000000	0.000000	0.000000
16	1.074000	1.075430	1.080250	1.081510	1.077080
17	1.075180	1.076350	1.080310	1.081450	1.077710
18	1.075460	1.076640	1.080350	1.085520	1.078020
19	1.075980	1.077030	1.080380	1.085040	1.078270
20	1.072420	1.074290	1.080180	1.082730	1.076430
21	1.076350	1.077270	1.080380	1.080890	1.078280
22	1.076920	1.077710	1.080420	1.080880	1.078590
23	1.075490	1.076630	1.080340	1.081740	1.077900
24	1.075840	1.076900	1.080360	1.081610	1.078080

Table A.14: The calculated stand-alone K_{eff}s using EVENT for Test Case 3 with a fuel inventory of 35 fuel elements in total, Part VII

Channel No.	Fuel 1	Fuel 2	Fuel 3	Fuel 4	Fuel 5
1	0.894511	0.892268	0.889881	0.883565	0.887277
2	0.979137	0.977504	0.975769	0.971158	0.973869
3	0.905068	0.903533	0.901898	0.897560	0.900111
4	0.929500	0.927332	0.925025	0.918959	0.922521
5	0.836266	0.834311	0.832236	0.826779	0.829980
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	1.079050	1.077610	1.076090	1.072010	1.074410
8	0.878916	0.877748	0.876507	0.873214	0.875149
9	1.056880	1.054460	1.051890	1.045110	1.049090
10	0.901310	0.899208	0.896978	0.891090	0.894546
11	1.124410	1.121990	1.119420	1.112620	1.116610
12	0.972672	0.970492	0.968184	0.962089	0.965666
13	1.197910	1.196590	1.195170	1.191460	1.193640
14	1.009170	1.008050	1.006850	1.003720	1.005560
15	0.000000	0.000000	0.000000	0.000000	0.000000
16	1.093040	1.090630	1.088060	1.081280	1.085270
17	1.005040	1.002740	1.000290	0.993819	0.997621
18	1.166010	1.164550	1.162990	1.158880	1.161290
19	1.092560	1.091140	1.089630	1.085640	1.087990
20	1.119170	1.117220	1.115130	1.109590	1.112850
21	0.960910	0.958463	0.955862	0.949019	0.953033
22	0.908363	0.905979	0.903449	0.896793	0.900699
23	1.026290	1.024140	1.021840	1.015740	1.019330
24	0.989159	0.987014	0.984723	0.978639	0.982217

Table A.15: The calculated stand-alone PPFs using EVENT for Test Case 3 with a fuel inventory of 35 fuel elements in total, Part I

Channel No.	Fuel 6	Fuel 7	Fuel 8	Fuel 9	Fuel 10
1	0.879704	0.876006	0.881852	0.877417	0.000000
2	0.968338	0.965630	0.969908	0.966666	0.000000
3	0.894905	0.892355	0.896382	0.893331	0.000000
4	0.915268	0.911747	0.917317	0.913087	0.000000
5	0.823455	0.820294	0.825303	0.821498	0.000000
6	0.000000	0.000000	0.000000	0.000000	1.000290
7	1.069510	1.067100	1.070900	1.068020	0.000000
8	0.871204	0.869270	0.872322	0.870009	0.000000
9	1.040970	1.037010	1.043270	1.038520	0.000000
10	0.887511	0.884091	0.889501	0.885396	0.000000
11	1.108480	1.104490	1.110780	1.106000	0.000000
12	0.958383	0.954847	0.960441	0.956193	0.000000
13	1.189190	1.187020	1.190450	1.187850	0.000000
14	1.001810	0.999997	1.002870	1.000690	0.000000
15	0.000000	0.000000	0.000000	0.000000	0.000000
16	1.077150	1.073210	1.079450	1.074710	0.000000
17	0.989882	0.986114	0.992072	0.987553	0.000000
18	1.156370	1.153970	1.157770	1.154890	0.000000
19	1.083200	1.080860	1.084560	1.081750	0.000000
20	1.106190	1.102910	1.108080	1.104160	0.000000
21	0.944865	0.940908	0.947172	0.942416	0.000000
22	0.892755	0.888906	0.894999	0.890368	0.000000
23	1.012000	1.008410	1.014080	1.009780	0.000000
24	0.974907	0.971324	0.976983	0.972692	0.000000

Table A.16: The calculated stand-alone PPFs using EVENT for Test Case 3 with a fuel inventory of 35 fuel elements in total, Part II

Channel No.	Fuel 11	Fuel 12	Fuel 13	Fuel 14	Fuel 15
1	0.000000	0.881095	0.871206	0.876303	0.874387
2	0.000000	0.970926	0.963499	0.967336	0.965897
3	0.000000	0.897241	0.890254	0.893864	0.892511
4	0.000000	0.917472	0.908073	0.912904	0.911082
5	0.000000	0.825433	0.816974	0.821319	0.819682
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	0.000000	1.071750	1.065090	1.068540	1.067250
8	0.000000	0.872388	0.867027	0.869797	0.868759
9	0.000000	1.043010	1.032400	1.037860	1.035810
10	0.000000	0.888905	0.879722	0.884446	0.882669
11	0.000000	1.110070	1.099570	1.104970	1.102940
12	0.000000	0.959662	0.950235	0.955084	0.953259
13	0.000000	1.189190	1.183440	1.186400	1.185290
14	0.000000	1.001580	0.996713	0.999220	0.998276
15	1.148910	0.000000	0.000000	0.000000	0.000000
16	0.000000	1.079440	1.068810	1.074280	1.072220
17	0.000000	0.992415	0.982230	0.987472	0.985501
18	0.000000	1.157830	1.151300	1.154670	1.153400
19	0.000000	1.084990	1.078590	1.081890	1.080650
20	0.000000	1.107210	1.098400	1.102960	1.101250
21	0.000000	0.947086	0.936545	0.941960	0.939919
22	0.000000	0.895145	0.884859	0.890142	0.888151
23	0.000000	1.013900	1.004250	1.009230	1.007360
24	0.000000	0.976976	0.967322	0.972309	0.970436

Table A.17: The calculated stand-alone PPFs using EVENT for Test Case 3 with a fuel inventory of 35 fuel elements in total, Part III

Channel No.	Fuel 16	Fuel 17	Fuel 18	Fuel 19	Fuel 20
1	0.868908	0.870177	0.879358	0.868963	0.867124
2	0.961765	0.962723	0.969627	0.961805	0.960414
3	0.888621	0.889523	0.896018	0.888660	0.887352
4	0.905907	0.907102	0.915813	0.905958	0.904230
5	0.815028	0.816101	0.823936	0.815074	0.813519
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	1.063520	1.064390	1.070590	1.063550	1.062300
8	0.865773	0.866464	0.871450	0.865801	0.864796
9	1.029940	1.031290	1.041150	1.030000	1.028030
10	0.877602	0.878772	0.887285	0.877650	0.875960
11	1.097140	1.098480	1.108220	1.097200	1.095260
12	0.948061	0.949261	0.958000	0.948110	0.946374
13	1.182110	1.182840	1.188180	1.182140	1.181070
14	0.995586	0.996208	1.000720	0.995612	0.994711
15	0.000000	0.000000	0.000000	0.000000	0.000000
16	1.066350	1.067710	1.077570	1.066410	1.064440
17	0.979871	0.981172	0.990622	0.979924	0.978043
18	1.149770	1.150610	1.156680	1.149810	1.148590
19	1.077090	1.077920	1.083870	1.077120	1.075930
20	1.096320	1.097470	1.105680	1.096370	1.094710
21	0.934122	0.935461	0.945225	0.934178	0.932247
22	0.882495	0.883799	0.893326	0.882549	0.880662
23	1.002000	1.003250	1.012210	1.002060	1.000260
24	0.965069	0.966315	0.975285	0.965121	0.963316

Table A.18: The calculated stand-alone PPFs using EVENT for Test Case 3 with a fuel inventory of 35 fuel elements in total, Part IV

Channel No.	Fuel 21	Fuel 22	Fuel 23	Fuel 24	Fuel 25
1	0.863359	0.865516	0.866246	0.857925	0.857442
2	0.957556	0.959194	0.959750	0.953413	0.953042
3	0.884664	0.886204	0.886725	0.880766	0.880420
4	0.900704	0.902721	0.903405	0.895663	0.895217
5	0.810358	0.812165	0.812780	0.805830	0.805430
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	1.059700	1.061190	1.061690	1.055900	1.055560
8	0.862729	0.863914	0.864315	0.859727	0.859461
9	1.024020	1.026320	1.027100	1.018250	1.017740
10	0.872505	0.874482	0.875153	0.867550	0.867115
11	1.091250	1.093520	1.094290	1.085560	1.085050
12	0.942828	0.944856	0.945547	0.937747	0.937298
13	1.178880	1.180130	1.180560	1.175720	1.175440
14	0.992868	0.993923	0.994282	0.990212	0.989974
15	0.000000	0.000000	0.000000	0.000000	0.000000
16	1.060430	1.062730	1.063510	1.054660	1.054150
17	0.974196	0.976399	0.977146	0.968673	0.968185
18	1.146080	1.147520	1.148000	1.142460	1.142140
19	1.073460	1.074880	1.075350	1.069890	1.069580
20	1.091280	1.093250	1.093910	1.086280	1.085830
21	0.928310	0.930561	0.931326	0.922684	0.922186
22	0.876825	0.879019	0.879767	0.871338	0.870852
23	0.996555	0.998676	0.999392	0.991198	0.990722
24	0.959607	0.961735	0.962453	0.954242	0.953763

Table A.19: The calculated stand-alone PPFs using EVENT for Test Case 3 with a fuel inventory of 35 fuel elements in total, Part V

Channel No.	Fuel 26	Fuel 27	Fuel 28	Fuel 29	Fuel 30
1	0.855093	0.862337	0.856182	0.861620	0.860423
2	0.951240	0.956780	0.952076	0.956234	0.955321
3	0.878727	0.883934	0.879510	0.883420	0.882562
4	0.893054	0.899752	0.894058	0.899086	0.897975
5	0.803493	0.809501	0.804392	0.808905	0.807905
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	1.053900	1.058990	1.054670	1.058490	1.057650
8	0.858154	0.862170	0.858759	0.861772	0.861110
9	1.015260	1.022930	1.016410	1.022170	1.020900
10	0.864987	0.871574	0.865970	0.870918	0.869823
11	1.082600	1.090170	1.083730	1.089420	1.088160
12	0.935117	0.941870	0.936126	0.941199	0.940078
13	1.174070	1.178280	1.174710	1.177870	1.177170
14	0.988828	0.992365	0.989360	0.992015	0.991431
15	0.000000	0.000000	0.000000	0.000000	0.000000
16	1.051670	1.059340	1.052810	1.058580	1.057310
17	0.965809	0.973159	0.966908	0.972427	0.971210
18	1.140560	1.145410	1.141290	1.144930	1.144130
19	1.068020	1.072800	1.068740	1.072330	1.071540
20	1.083640	1.090340	1.084660	1.089690	1.088580
21	0.919775	0.927244	0.920891	0.926500	0.925261
22	0.868504	0.875788	0.869594	0.875059	0.873851
23	0.988398	0.995551	0.989477	0.994844	0.993664
24	0.951433	0.958602	0.952515	0.957895	0.956713

Table A.20: The calculated stand-alone PPFs using EVENT for Test Case 3 with a fuel inventory of 35 fuel elements in total, Part VI

Channel No.	Fuel 31	Fuel 32	Fuel 33	Fuel 34	Fuel 35
1	0.854106	0.859358	0.877762	0.865680	0.865346
2	0.950483	0.954507	0.968433	0.961700	0.959178
3	0.878015	0.881796	0.894896	0.888345	0.886189
4	0.892150	0.896987	0.914295	0.904390	0.902497
5	0.802683	0.807020	0.822569	0.813610	0.811969
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	1.053210	1.056910	1.069520	1.062640	1.061320
8	0.857604	0.860519	0.870589	0.863917	0.864029
9	1.014220	1.019770	1.039430	1.026500	1.026200
10	0.864097	0.868852	0.885801	0.873910	0.874383
11	1.081580	1.087050	1.106520	1.092960	1.093310
12	0.934203	0.939081	0.956476	0.943989	0.944681
13	1.173500	1.176550	1.187250	1.178060	1.180030
14	0.988347	0.990911	0.999937	0.991610	0.993868
15	0.000000	0.000000	0.000000	0.000000	0.000000
16	1.050630	1.056180	1.075850	1.063380	1.062610
17	0.964813	0.970126	0.988977	0.977595	0.976309
18	1.139910	1.143420	1.155630	1.147470	1.147590
19	1.067370	1.070840	1.082840	1.075430	1.074980
20	1.082720	1.087600	1.104260	1.091840	1.093110
21	0.918768	0.924157	0.943517	0.931971	0.930322
22	0.867524	0.872778	0.891662	0.880778	0.878798
23	0.987425	0.992615	1.010660	1.000070	0.998572
24	0.950456	0.955658	0.973732	0.963419	0.961650

Table A.21: The calculated stand-alone PPFs using EVENT for Test Case 3 with a fuel inventory of 35 fuel elements in total, Part VII

Appendix B

Fast K_{eff} Prediction for the CONSORT reactor using Artificial Neural Networks

A feed forward multi-layer perceptron Artificial Neural Network (ANN) was built for predicting the K_{eff} for the CONSORT reactor. Each test case has its own ANN trained. The ANN provides the LP evaluation for our EDAs and GAs instead of EVENT.

For Test Case 1, the ANN constructed has 24 input nodes representing the fuel channels, 65 hidden nodes and 1 output node. A total of 1535 different LPs are generated randomly, their K_{eff} values are calculated by EVENT. A subset of 767 of LPs were used for training, 460 for validation and 308 for testing. The ANN is generated and trained using the Stuttgart Neural Network Simulator (SNNS, Zell et al. (1995)) package. Using a trained ANN developed here, K_{eff} of 1000 LPs can be predicted within a second with errors of 0.19% against EVENT calculations on average, with approximately 450,000 nodes in the finite element model. Meanwhile, the computational time needed for the LP evaluation can be reduced massively.

The inputs for our ANN are similar to those used by Erdogan and Geckinli (Erdogan and Geckinli (2003)). Instead of using the fuel element identifiers in integer

form, we use the square of the normalised stand-alone K_{eff} with fuel coupling. In Test Case 1, for a MARK III fuel element (fuel type '5') in channel 1, the input to the ANN will be given by the square of the corresponding entry [1, 5] in the stand-alone K_{eff} table 7.3, which is 1.20883. For example, a sub-LP X from channel 1 to channel 6:

$$X = [5, 1, 3, 5, 2, 4] \tag{B.1}$$

which means channels 1, 2, 3, 4, 5 and 6 are loaded with MARK III, MARK I_A, MARK I_C, MARK III, MARK I_B and MARK II, respectively. Applying the square of stand-alone K_{eff}, X can be transformed to:

$$X' = [1.20883^2, 1.20462^2, 1.196542^2, 1.207352^2, 1.201642^2, 1.210912^2] \tag{B.2}$$

By doing this, the variance is reasonably rescaled and therefore helps the ANN to recognize different loading patterns more precisely. Then all the entries in X' are normalized between [0.2, 0.8] according to the following equation:

$$\frac{X'_i - LowerBound}{UpperBound - LowerBound} \cdot 0.6 + 0.2, i = 1, 2, ..., 6 \tag{B.3}$$

where the $LowerBound$ and the $UpperBound$ are the square of the minimum and the maximum entries in the stand-alone K_{eff} table, which are $1.169352 = 1.3674$, and $1.219722 = 1.4877$, at [15, 3] and [15, 5], respectively. Therefore the final input vector for the ANN is:

$$X'' = [0.6683, 0.6174, 0.5208, 0.6504, 0.5818, 0.6934] \tag{B.4}$$

The corresponding K_{eff}s are normalized between [0.2,0.8] which are the expected output of the ANN. We use [0.2, 0.8] instead of [0, 1] to ensure the training data are not too close to the output boundary of the ANN, which is [0, 1].

This ANN is tested using the training LP set and the unseen test LP sets. The

word ‘unseen’ means the LPs in the test set are not included in the training and the validation set. The average prediction error, the largest prediction error, number of error ranges within 1% and 0.5% are calculated. The error measurement is defined below:

$$Error = \frac{|\text{ANN's Prediction - EVENT Calculation}|}{\text{EVENT Calculation}} \tag{B.5}$$

All of the ANNs prediction test results are shown in Chapter 7.

Appendix C

The 2-Point Crossover used in a Benchmark GA

The 2-point crossover (2PX) operator used in the GAs is one of the earliest and most commonly used crossover operators. It works with a wide range of encoding methods, including binary string, real value vector and integer vector, most likely with a 1D structure. It simply breaks and recombines the encoded string randomly to generate new solutions.

In Test Case 1, the LP representation is an integer vector of fuel type identifiers indexed by the core channel number. The 2PX can be applied naturally. For example, a sub-LP giving the loading from core channel 1 to channel 6 is:

$$X = [5, 1, 3, 5, 2, 4] \tag{C.1}$$

which means channels 1 to 6 are loaded with MARK III, MARK I_A, MARK I_C, MARK III, MARK I_B and MARK II, respectively. A complete LP is such a vector, but its length is 24, the number of fuel channels. Each element is an integer in [1,5] indicating a fuel type.

As shown in Figure C.1, the basic steps of 2-point crossover are as follows. Firstly, choose two LPs as parents, then choose two points in the encoded string randomly to perform crossover. The last step is to generate new LPs by swapping

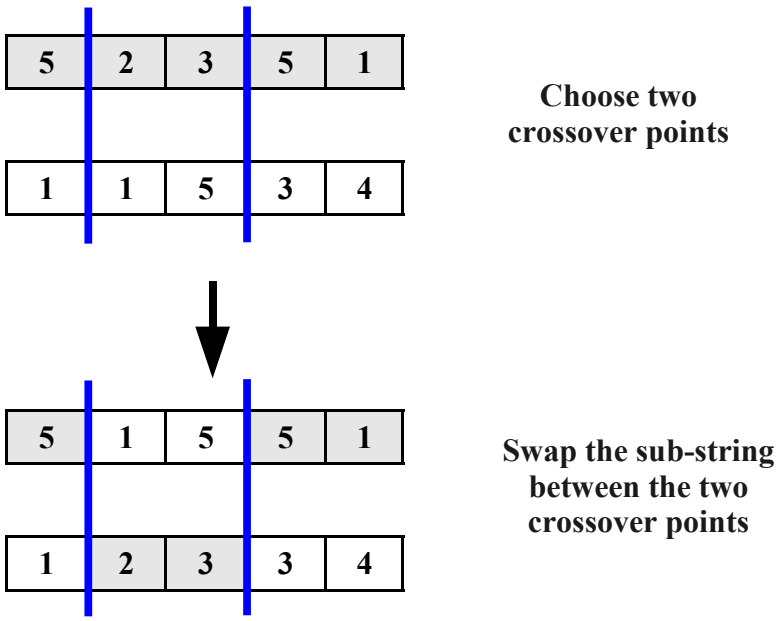


Figure C.1: An example of the two-point crossover operator used in the benchmark GA

the sub-pattern between the two points.

Since the basic 2PX cannot be applied to permutation encoding, GA_2PX is not used as a benchmark for Test Case 2 and 3.